The Design, Implementation and Analysis of a Wavelet-Based Video Codec

by

Marc Paul Servais

Submitted to the Department of Electrical Engineering
in partial fulfilment of the requirements for the degree of

Master of Science in Engineering

at the

UNIVERSITY OF CAPE TOWN

October 1998

Declaration

I, Marc Paul Servais, declare that this dissertation is my own work. It is being submitted for the degree of Master of Science in Engineering at the University of Cape Town and it has not been submitted before for any degree or examination, at any other university.

Marc Paul Servais

Acknowledgements

I would like to thank my supervisor, Professor Gerhard de Jager, for his enthusiasm and guidance throughout this project.

I also wish to thank:

- Dr. Brendt Wohlberg for providing much-needed assistance with aspects of wavelet theory, and for helping to review a draft of this report.
- My parents, brothers and sister for their encouragement and prayers.
- Telkom S.A. Ltd. for their generous financial support.
- All the members of the UCT Digital Image Processing Lab, for providing a stimulating working environment. In particular, thanks to Willie Nel, Fred Nicolls and Praven Reddy.

Abstract

The Wavelet Transform has been shown to be highly effective in image coding applications. This thesis describes the development of a new wavelet-based video compression algorithm which is based on the 3D wavelet transform, and requires no complicated motion estimation techniques.

The proposed codec processes a sequence of images in a group of frames (GOF) by first transforming the group spatially and temporally, in order to obtain a GOF of 3D approximation and detail coefficients. The codec uses selective prediction of temporal approximation coefficients in order to decorrelate transformed GOFs. Following this, a modified version of Said and Pearlman's image coding technique of Set Partitioning in Hierarchical Trees is used as a method for encoding the transformed GOF.

The compression algorithm has been implemented in software, and tested on seven test sequences at different bit-rates. Experimental results indicate a significantly improved performance over MPEG 1 and 2 in terms of picture quality, for sequences filmed with a stationary camera. The codec also performs well on scenes filmed with a moving camera, provided that there is not a large degree of spatial detail present.

In addition, the proposed codec has several attractive features. It performs well without entropy coding, and does not require any computationally-expensive motion estimation methods, such as those used by MPEG. Finally, a substantial advantage is that the encoder generates a bit-stream which allows for the progressive transmission of video, making it well-suited to use in video applications over digital networks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "The biggest obstacle to the vaunted multimedia revolution is digital obesity. That's the bloat that occurs when pictures, sound and video are converted from their natural analog form into computer language for manipulation or transmission. ... Compression, a rapidly developing branch of mathematics, is putting digital on a diet. ... Its popularity is rooted in economics: compression lowers the cost of storage and transmission by packing data into a smaller space. Many new electronic products and services simply couldn't exist without it."
>
> *Business Week, 14 February 1994*

1.1 Background

The growth in digital technology over the past couple of decades has impacted substantially on multimedia services. Sound, images and video can now be stored and transmitted without any loss in quality. Services such as tele-medicine, distance education, video-conferencing and Internet retailing are currently used by relatively few people, but are rapidly gaining in popularity.

During the same period, there have also been significant advances in network technology. Broadband Networks have been developed to allow for large amounts of data to be transmitted at high rates. Advances in optical fibre technology, and the development of improved modulation techniques have helped to provide more resources for the transfer of information.

Optical storage techniques have revolutionised the storage of digital information, and are now in widespread use. For example, a compact disc (CD) can store 650 megabytes of data or more than one hour of high-quality stereo audio.

However, the *demand* for multimedia services has grown at a substantially greater rate than advances in transmission or storage technology. This problem is compounded by the vast amounts of data associated with multimedia applications. Digital video, in particular, requires huge amounts of data for storage. For example, the amount of digital memory required to store an entire copy of the Bible, can store only *one quarter of a second* of television-quality video.

In practice, the compression of digital video data is essential for the vast majority of video-based applications. The primary goal in image or video compression is to represent the data as efficiently as possible, without a noticeable loss in image quality.

1.2 Thesis Objectives

This report describes the design, implementation and analysis of a video codec. The only initial requirement was that the codec be based on the *wavelet transform*, which had proven to be successful in image coding. Additional requirements for the codec were only defined during the initial stages of the project, once a review of the appropriate literature had been completed. The design objectives are listed in Section 2.4.3.

1.3 Structure of Report

This remainder of this dissertation is structured as follows:

Chapter 2 provides an introduction to compression. It highlights the need for the compression of image and video data, and gives a brief overview of the JPEG [21] and MPEG [6] standards. Following this, some aspects of human vision are described - in particular, the precision with which an image must be displayed to be of reasonable quality to a human observer. Finally, some performance characteristics of video codecs are discussed, and the Design Objective for the proposed codec is presented.

Chapter 3 presents an overview of wavelet theory. It describes wavelets as basis functions which are *localised* in both time/space and frequency, and discusses the advantages of this for signal compression. The concept of approximating a signal at various scales using *multiresolution analysis* is also presented. Following this, there is an overview of the implementation of the wavelet transform using filter banks. Finally, the role of the transform in achieving *compression* is discussed.

Chapter 4 introduces the 3D wavelet transform and describes how it may be applied to video. The concepts of *prediction* and *selective prediction* of (temporal) wavelet approximation coefficients are also examined. These are presented as a means of improving the compression performance of a video codec. Finally, the above ideas are extended to colour video sequences.

Chapter 5 describes how wavelet coefficients may be described *efficiently*, so as to enable effective compression. The concept of *progressive transmission* is also discussed, and its advantages (particularly those relating to the *transmission* of video) are highlighted. A new extension of SPIHT[1] [25], developed for the coding of *video*, is presented. Finally, a brief overview of entropy coding is provided.

Chapter 6 describes the integration of the above techniques (the 3D wavelet transform and modified SPIHT) into a new video codec, *SPRED*. The wavelet-based SPRED codec uses **S**et **P**artitioning and the **S**elective **Pred**iction of **DC** frames. The structures of the encoder and decoder are described in detail.

Chapter 7 demonstrates the performance of SPRED relative to MPEG-1 and MPEG-2. Comparative results are shown for seven test video sequences at two bit-rates. The progressive transmission capabilities of SPRED are also shown, as well as its performance *without* entropy coding.

Chapter 8 concludes this report, stating the major findings along with several recommendations.

Appendix A provides mappings for RGB to YC_bC_r (and vice versa) colour space conversions.

Appendix B illustrates detailed results not included in Chapter 7.

Appendix C provides the parameters used for compressing the test sequences with MPEG-1 and MPEG-2.

The CD-ROM lists the SPRED source code. In addition, it contains some of the sequences (in *uncompressed AVI* format) referred to in Chapter 7 . Refer to the "README.txt" file for additional details.

[1] SPIHT has been demonstrated to be an effective *image* coding scheme [25].

Chapter 2

Video Compression: *Setting the Scene*

2.1 The Need for Video Compression

Over the past decade, significant advances in digital multimedia technology have resulted in the increased use of digital video as a means of communication. Digital satellite broadcasting, video-on-demand and video-conferencing are some of the more common applications of digital video.

However, the above services, while readily available in many parts of the world, are by no means in common use. This is largely due to the cost and resource requirements of digital video applications. For example, to transmit *uncompressed* television quality video[1] would require transmission at a bit-rate in excess of 150 Mbps! Similarly, a CD-ROM (which can store *74 minutes* of uncompressed stereo audio in digital format) can only store *32 seconds* of uncompressed television quality video.

The above examples illustrate two consequences of the fact that digital video consists of voluminous amounts of data. This poses significant problems for the transmission, manipulation, storage and retrieval of digital video data. The need for a more efficient representation of digital video is thus self-evident.

Video Compression refers to the process of encoding digital video data in an efficient way by eliminating redundant information. In *lossless* video compression, digital video data is compressed (encoded) in such a way that the original video signal may be perfectly reconstructed. In *lossy* video compression, the compressed data may be used to reconstruct an *approximate*

[1] Assuming PAL format: 720 pixels x 576 pixels/line, 16 bit colour and 25 frames per second.

version of the original video signal. Consequently, some distortion is introduced during lossy compression. As will be shown in Section 2.4.2, this may allow for greater compression.

2.2 Compression Standards

2.2.1 JPEG

The *Joint Photographic Experts Group* (JPEG) was established in 1986. The group was tasked with formulating a standard [21] for the compression of continuous-tone still images, in both greyscale and colour. The JPEG standard[2] is now widely used for digital image compression, and is particularly popular on the World Wide Web. The JPEG committee is currently investigating improvements to the current standard, and is due to release a new standard in 2000. It is widely believed that the forthcoming *JPEG-2000* standard will be based on the *wavelet transform*, which has shown a significant improvement in performance over DCT-based methods [25, 28].

2.2.2 MPEG

In 1988, the *Moving Picture Experts Group* (MPEG) was established by the International Standards Organisation (ISO) with a mandate to establish standards for the coded representation of digital video and audio. To date, MPEG has produced two standards:

- MPEG-1 (finalised in 1991) was designed for the storage and retrieval of digital video and associated audio. It was optimised for non-interlaced [19] video of *Source Input Format* (SIF) resolution[3] at bit-rates of around 1.1 Mega-bits per second (Mbps)[4] for the video component. However, higher resolution and bit rates are possible [19].
- MPEG-2 (finalised in 1994) was developed as the standard for digital television. Consequently, it has features to handle frame interlacing. It was optimised for television quality video[5] at a bit rate of 4 Mbps [19], though it performs well over a wide range of bit-rates.

As with JPEG, MPEG 1 and 2 both use the Discrete Cosine Transform for the coding of *intra* frames. MPEG 1 and 2 also use block-based motion estimation techniques to calculate motion vectors. Motion vectors provide an effective means of describing *translational* motion between

[2] JPEG is based on the *Discrete Cosine Transform* (DCT)[24].
[3] 352 × 240 pixels, 30 frames/second (NTSC) or 352 × 288 pixels, 25 frames/second (PAL).
[4] 1 Mbps is equivalent to 2^{20} or 1,048,576 bits per second.
[5] Television quality video is about twice SIF resolution, both horizontally and vertically.

successive frames. However, the process of calculating motion vectors can be computationally intensive. In addition, motion vectors do not adequately model other forms of motion such as object rotation.

The MPEG committee is currently in the process of developing two more standards:[6]

- MPEG-4 (due to be finalised in December 1998) is a standard for multimedia applications. Fundamental to MPEG-4 is the notion of *media objects*. Thus in a video sequence of a woman talking, the moving image of the particular woman constitutes a media object, as does her voice. In addition, the moving image of the woman *and* her voice together constitute a *compound media object* [6].
- MPEG-7 (due to be finalised in November 2001) is not directly related to video compression. It is intended for multimedia content representation, i.e. describing the content of multimedia data. This will allow for efficient multimedia search engines, similar to current text-based search engines [6].

2.3 Aspects of Human Vision

From an image observer's point of view, it is not necessary for digital pictures to contain detail which is not perceptible to the intended observer. In the arena of visual communications, virtually all digital images and videos are intended for human viewers.

Consequently, images need only contain detail which can be perceived by the human eye, or more precisely the *Human Visual System* (HVS). The HVS consists of the eyes, several parts of the brain and the nerve fibres connecting them. The eye contains two systems, one for forming the image and the other for transducing the image into electrical impulses [2].

This section examines the sensitivity of the HVS to spatial and temporal resolution and to the perception of colour. By knowing this, it is possible to eliminate redundant[7] information from an image, so as to improve the compression achieved by a video compression codec.

2.3.1 Spatial Sampling

A digital image can be considered as a collection of regular samples of the intensity of some scene. Such samples are commonly referred to as pixels.[8] For a high-quality representation

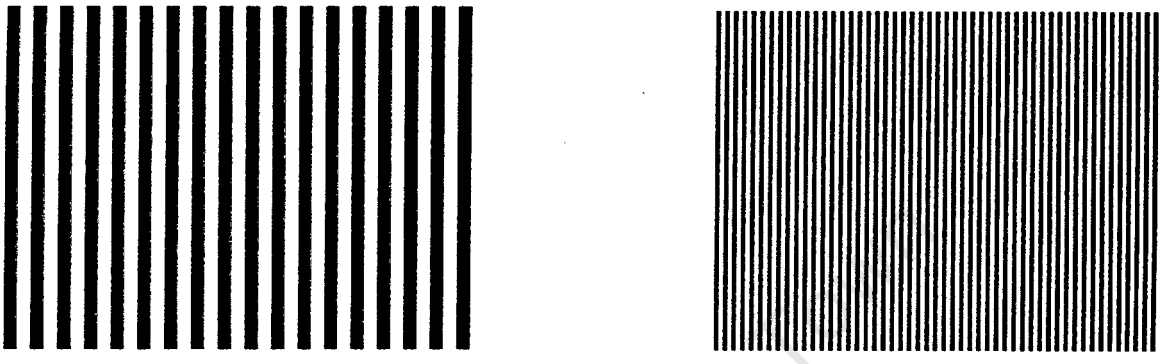
[6]Interestingly, the numbering of the MPEG standards (1,2,4 and 7) does not follow any particular sequence, though the larger numbers correspond to the more recent standards [6].

[7]From the point of view of a human observer.

[8]In a greyscale image, each sample constitutes a pixel. In a typical colour image, a pixel consists of three samples, one from each colour band.

of the scene, the minimum number of samples needed in the digital image depends on the response of the HVS to changes in intensity.

Figure 2.1 depicts two gratings with different spatial frequency components. The HVS is more easily able to distinguish individual lines in the lower frequency grating of Figure 2.1(a) than in the higher frequency grating of Figure 2.1(b).



(a) (b)

Figure 2.1: Gratings at different spatial frequencies.

The curve labelled "Luminance" (which roughly corresponds to brightness) in Figure 2.2 shows the sensitivity of the HVS to changes in intensity over a range of spatial frequencies. Note that the graph peaks at a spatial resolution of about 5 cycles/degree. This implies that at a viewing distance of a metre, for example, the HVS would be most sensitive (in terms of intensity changes) to objects with a thickness of roughly 2 mm [21].

From the graph, it is evident that the sensitivity of the HVS drops off significantly at higher spatial frequencies. Thus the high frequency content of an image can be described somewhat more coarsely than the lower frequency information, without the difference being noticeable to a human observer.

Sample Precision

In addition to the density of samples in an image, the precision with which pixels are specified is also an important requirement for accurate representation of a scene. Hunt [9] has shown that the HVS is able to distinguish about 100 different luminance levels in uniform-perceptual spaces. In practice, colour spaces are not linear, and 8 bits are used per sample. This corresponds to a possible 256 distinct values per luminance sample.

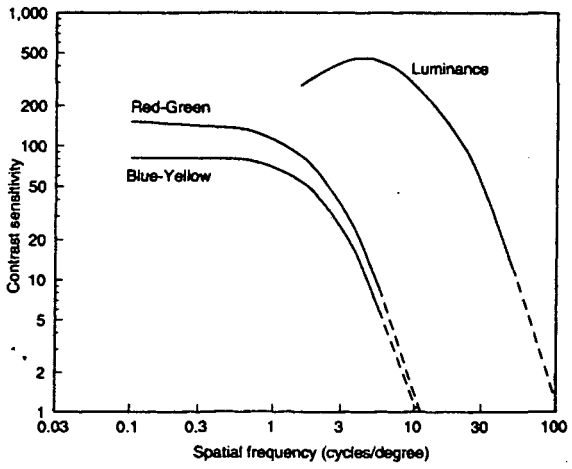


Figure 2.2: Sensitivity of the eye to luminance and chrominance intensity changes. From *Pennebaker and Mitchell* [21, p. 24]

2.3.2 Colour Vision

According to the *trichromatic theory of colour vision*, the eye has three types of cones (i.e. receptors) for colour [2]. Each type of cone is sensitive to a wide range of colours[9], but is most responsive in a short range of wavelengths. According to the theory, the three types of cones are particularly responsive to red, green-yellow and blue light.

Mullen [18] conducted a series of experiments to determine the sensitivity of the HVS to spatial frequency for coloured images. As depicted in Figure 2.2, the response of the HVS in terms of sensitivity to spatial frequency is significantly different for coloured light. From the graph, it is evident that the HVS is much less sensitive to detail in Red-Green and Yellow-Blue light than in white light of the same intensity.

From a compression point of view, it is therefore possible to have fewer colour[10] samples than luminance samples in an image, without introducing distortion that is visible to a human viewer.

Colour Spaces and Sub-sampling

Colour digital images are represented by numbers (samples) in a colour space. This section provides a brief overview of two commonly used colour spaces, namely RGB and YC_rC_b [19, 21].

[9] With wavelengths in the visible spectrum of 400 to 700 nm.
[10] Or more precisely, *chrominance*.

As mentioned above, the eye is believed to have three types of receptors which are most responsive to red, green and blue light. Consequently, the RGB[11] colour space is widely used for display purposes (e.g. in the television industry). Each pixel displays varying amounts of red, green and blue light in order to generate the desired colour and intensity.

A widely used colour scheme in image and video compression is the YC_rC_b colour space.[12] In this space, the *luminance* component (Y) is distinct from the two *chrominance* components (C_r and C_b). This separation allows for the implementation of *chrominance sub-sampling*, since (as discussed above) the HVS is relatively insensitive to high spatial frequencies in chrominance samples. Figure 2.3 gives an example of *4:2:0* chrominance sub-sampling, where the resolution of each of the chrominance components (C_r and C_b) is half that of the luminance component (Y), both horizontally and vertically.

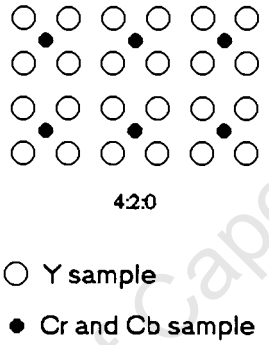


Figure 2.3: An illustration of 4:2:0 Chrominance Sub-sampling.

Appendix A shows the details for converting from the RGB to the YC_rC_b colour space, and vice versa.

2.3.3 Temporal Sampling

A video consists of a sequence of images, displayed in rapid succession, to give an appearance of continuous motion. If the time gap between consecutive frames it too large, the eye will observe jerky motion. From a video compression perspective, it is important not to transmit any redundant information. Thus the frame rate of a video should be as low as possible, without causing any significant distortion in motion, as perceived by a human observer.

Figure 2.4 shows the sensitivity of the HVS to *temporal* frequency as well as *spatial* frequency. From this graph it is evident that the sensitivity of the HVS begins to drop off significantly

[11] Red, Green and Blue.
[12] YC_rC_b is a variant of the YUV colour space.

at frame rates above about 24 frames per second (fps). In motion pictures, the frame rate used is 24 fps, while in television it is 25 fps (for PAL) or 30 fps (for NTSC). [13]

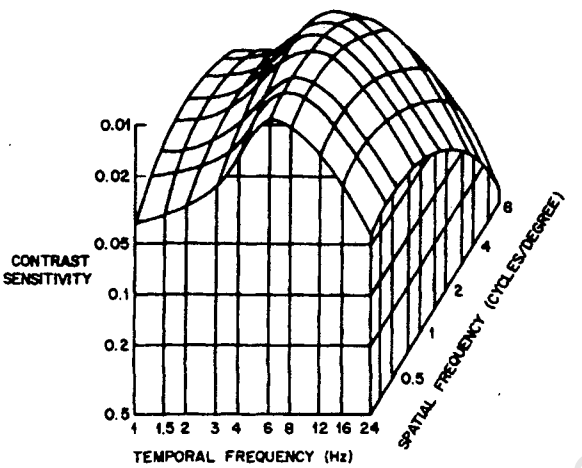


Figure 2.4: A Perspective view of the spatio-temporal threshold surface. From *Netravali and Haskell* [19, p. 280]

2.4 Video Codec Design

2.4.1 Performance Characteristics of a Video Codec

A *video codec* is an algorithm (often implemented in software) which performs compression and decompression of digital video. This section provides an overview of some performance criteria of video codecs.

Picture Quality

Picture quality is a measure of how well a compressed (and subsequently decompressed) image resembles the original (uncompressed) image. Quality can be measured either subjectively (by a human viewer or a group of human viewers) or objectively (using an appropriate mathematical formula).

A problem with using human observers to judge quality is that it is a time-consuming process. Furthermore, results may be influenced by the test environment and bias in the subjects. On the other hand, an objective method is consistent and generally easy to implement. However, results obtained from objective testing may not correlate well with a human understanding of picture quality.

[13] Note that with *interlacing* there are two *fields* per frame.

Probably the most commonly used method of measuring picture quality objectively is to calculate the *Peak-Signal to Noise Ratio* (PSNR) between an original and a compressed image. Note that for video sequences, the PSNR is calculated for each frame in the sequence. For video, this is often averaged to give the mean PSNR for a sequence. For an original and a compressed frame, the PSNR is calculated according to the formula:

$$\text{PSNR (in dB)} = 10 \log_{10} \left(\frac{X_{max}^2}{\text{MSE}} \right)$$

where X_{max} is the maximum possible intensity in the input image (e.g. 255 for a sample precision of 8 bits) and the *Mean Square Error* (MSE) is given by:

$$\text{MSE} = \frac{1}{N_R N_C} \sum_{i=1}^{N_R} \sum_{j=1}^{N_C} [X(i,j) - Y(i,j)]^2$$

where the number of rows and columns in the image is N_R and N_C respectively. $X(i,j)$ represents the intensity of a pixel with co-ordinate (i,j) in the original image. $Y(i,j)$ refers to the intensity of the corresponding pixel in the compressed (and subsequently decompressed) image.

Degree of Compression

The extent to which a digital image or video is compressed is commonly expressed in either relative or absolute terms. A *compression ratio* is a measure of the size of a compressed file, relative to the size of the original (uncompressed) file. Alternatively, the size of a (compressed) image or video file may be expressed in *bits per pixel* (bpp).

The extent to which a video is compressed is often specified by the resultant *bit-rate*, which is measured in bits per second (bps). This measure is particularly useful when the compressed bit-stream is intended for transmission (as opposed to storage in a file). Note that the number of bits per second is also influenced by the the video resolution and frame rate. Consequently, the bit-rates of the uncompressed and compressed video data should be compared, in order to measure the compression performance of a codec.

Additional Features

Several other features in video codecs are slightly less important, but still desirable. The *complexity* of a video codec is an important factor in many applications. It incorporates factors such as memory usage, execution time and ease of implementation.

For multicasting applications over networks, some decoders may only be able to decode at a lower resolution or frame than that at which the original video was encoded. A codec is considered to be spatially and/or temporally *scalable* if pre-defined portions of the bit-stream generated by the encoder may be decoded to produce a version of the original sequence at a lower resolution and/or frame rate.

The bit-stream produced by an encoder is *rate scalable* if it allows for the *progressive transmission* of video. This implies that even if only *part* of a bit-stream is processed by the decoder, a lower-quality version of the encoded sequence can be decoded (though not necessarily at a lower resolution or frame rate).

In many transmission channels, data can be corrupted, thus resulting in errors. It is therefore desirable that a compressed bit-stream be relatively insensitive to errors. In addition, if an error *does* occur in part of a bit-stream (and subsequently produces some image distortion at the decoder), the propagation of such distortion should (ideally) be *limited* so that it does not affect the image quality of subsequent frames.

2.4.2 The *Compression-Quality-Complexity* Tradeoff

In video compression there is a tradeoff between the degree to which an image sequence may be compressed, and the resultant quality of the compressed video. However, the other factors listed above are also significant. For example, it may be possible to design a codec which achieves substantial compression of a certain class of images, while still maintaining very good image quality. However, it may be that such a codec is incredibly complex and requires minutes to process a frame. Clearly such a codec would be unsuitable for practical applications.

Consequently, many codecs have been designed for specific applications in such a way that the performance of the codec (in terms of picture quality, compression ratios, efficiency, etc.) has been optimised for that particular application. For instance, a video-on-demand codec would need to be very efficient at decoding, but not necessarily for encoding.

Some codecs have been designed for specific types of scenes. For example, H.261 [26] and H.263 [27] are video-conferencing codecs which are intended primarily for "head and shoulder" scenes with a fixed camera and relatively small amounts of motion.

2.4.3 The Design Objective

During the initial stage of the design process, several requirements and guidelines for the proposed video codec were established. It was required that the proposed video codec:

- be based on the 3D wavelet transform (since the 2D transform had proved successful in image compression applications);
- not use any computationally expensive motion estimation techniques;
- not permit the propagation errors for more than some specified period (which may be provided by the user);
- generate compressed bit-streams which are *rate scalable*; and
- be able to compress both greyscale and colour "natural" image sequences.

Due to project time constraints, it was *not* required that the *software implementation* of the codec be optimised for speed.

The next chapter presents an overview of wavelet theory from a signal compression perspective. Following this, the principles behind the proposed *SPRED* codec are discussed.

Chapter 3

The Theory of Signal Compression Using the Wavelet Transform

The field of *wavelets* has developed significantly over the last decade. Wavelets have been applied in areas ranging from stock market prediction to the study of the distant universe. This chapter gives an introduction to wavelets and focuses on their application in the arena of signal compression.

3.1 Signals and Basis Functions

Consider a one-dimensional signal $f(t) \in L^2(\mathbb{R})$ which varies with time.[1] [2] It can be represented as a sum of *basis functions*:

$$f(t) = \sum_i c_i \Psi_i(t)$$

where $\{\Psi_i(t)\}_{i\in\mathbb{Z}}$ are the basis functions and $\{c_i\}_{i\in\mathbb{Z}}$ are the coefficients with which each basis function is weighted. In *Fourier Analysis* the basis functions used are sines and cosines, in which case the $\{c_i\}$ correspond to the Fourier coefficients of the signal.

[1] $L^2(\mathbb{R})$ is the set of all real-valued functions of $\mathbb{R}$ which are square integrable, i.e.

$$L^2(\mathbb{R}) = \left\{ g(x) \mid g : \mathbb{R} \to \mathbb{R}, \int g(x)^2 \, dx < \infty \right\}$$

[2] In this discussion the signal is defined in time. However, one could equally consider a signal in space, $f(x)$ say.

It is interesting to note that $f(t)$ directly conveys information about the signal's behaviour in time, but not in frequency. On the other hand, the Fourier Domain representation describes the function in terms of its constituent frequencies, but gives no indication of the Time Domain behaviour of the function.

In signal compression it is often desirable to have information about the frequency of a signal over a specific region or time interval [7]. This is particularly relevant for many "natural" signals with periods or regions of high correlation, such as a sample of sound or an object in an image. Low-frequency signals are generally defined over a large interval, as in the case of an object with fairly constant intensity. In contrast, high frequency signals typically occur over short intervals, such as a transition in intensity at an object boundary.

Wavelets are basis functions which can be used to achieve such a function decomposition, since they are *localised* in both time and frequency. In additional, the basis functions, $\{\Psi_i(t)\}$, may take on various shapes. Thus an appropriate set of basis functions may be chosen if they are more suited to a particular application.[3]

The Discrete Wavelet Transform

The discrete wavelet transform of a function, $f(t) \in \mathbb{R}$, is defined in terms of inner products by the equation:

$$d_{i,j} = \langle \psi_{i,j}, f \rangle \tag{3.1}$$

where $d_{i,j}$ are the detail coefficients. The wavelet basis functions (at resolution i) are given by $\psi_{i,j}(t)$. This is a *shifted* and *scaled* version of the *mother wavelet*, $\psi(t)$.

The function, $f(t)$, may then be represented with a desired accuracy, using a sufficient number of detail coefficients. The inverse wavelet transform is given by the equation:

$$f = \sum_i \sum_j d_{i,j} \psi_{i,j} \tag{3.2}$$

3.2 Multiresolution Analysis

Multiresolution Analysis (MRA) is a procedure developed by Mallat [14] for the construction

[3]For example, a basis function which represents edges well may be appropriate for image compression.

of dyadic[4] wavelets. This section provides a brief overview of some of the basic principles of MRA.

3.2.1 Approximation and Detail Spaces

An *Approximation Space* is a subspace of $L^2(\mathbb{R})$ in which a function may be described with a specific resolution. Figure 3.1 depicts the same function in four approximation spaces. An approximation space is generally represented by the notation V_i, where i gives an indication of the resolution with which the function is approximated.[5]

A *Detail Space* is simply the information in an approximation space which *cannot* be represented in an approximation space of of *lower* resolution. Thus the detail space, W_i, is defined by the equation:[6]

$$V_i \oplus W_i = V_{i+1} \tag{3.3}$$

3.2.2 Mallat's MRA

A multiresolution analysis of $L^2(\mathbb{R})$ is a sequence of closed subspaces $\{\dots, V_{-1}, V_0, V_1, \dots\}$ which fulfill the following requirements:

1. $V_i \subset V_{i+1}$ (Containment)

This requirement specifies that a function in approximation space V_i can be described in a higher resolution space V_{i+1} without a loss of detail.

2. $\overline{\bigcup_{i\in\mathbb{Z}} V_i} = L^2(\mathbb{R})$ (Upward completeness)

3. $\bigcap_{i\in\mathbb{Z}} V_i = \{0\}$ (Downward completeness)

The above two equalities indicate that a function may me approximated at an arbitrarily high or low resolution. This allows for any function in $L^2(\mathbb{R})$ to be expressed as the sum of components from *detail* spaces, indexed over all integers.

4. $f(x) \in V_i \Leftrightarrow f(2x) \in V_{i+1}$ (Scale invariance)

5. $f(x) \in V_0 \Leftrightarrow f(x+1) \in V_0$ (Shift invariance)

Statements 4 and 5 refer to approximation spaces. However, *detail spaces* inherit their struc-

[4]Dyadic wavelets satisfy the form, $\psi_{i,j} = \sqrt{2^i}\psi(2^i x - j)$

[5]A larger value of i indicates a higher resolution.

[6]$A \oplus B$ represents the *orthogonal sum* of subspaces A and B.

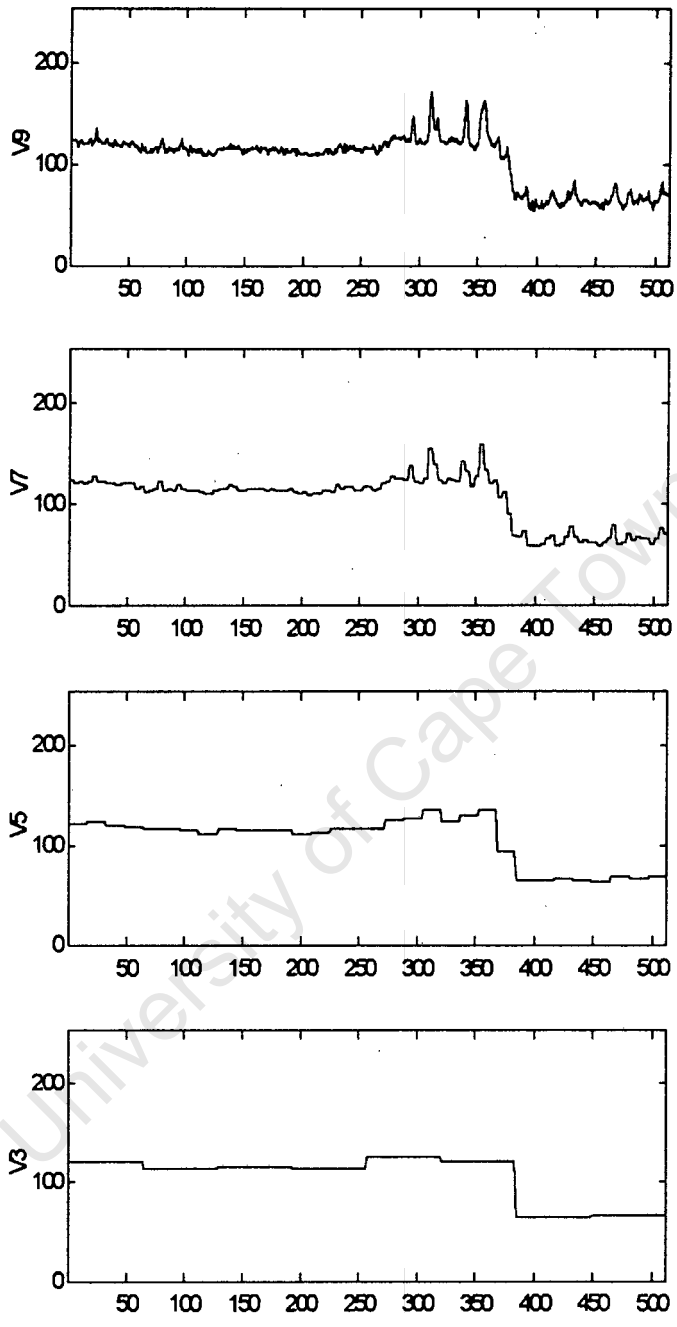


Figure 3.1: A function depicted in four approximation spaces with decreasing resolution: V_9, V_7, V_5 and V_3 (using the Haar basis). The original function is the luminance values of a column of 512 pixels of the *Lena* image, representing a vertical cross-section of the image.

ture from approximation spaces (as indicated in Equation 3.3). Consequently, implication 4 ensures that the basis of each detail space is the basis of the *previous* detail space, dilated by a factor of two. Statement 5 implies that each set of basis functions for a detail space may be obtained by the translation of a single function. This function is commonly referred to as the *wavelet*, $\psi(x)$.

6. $\exists \phi \in V_0$ such that $\int_{-\infty}^{\infty} \phi(x)\,dx \neq 0$ and $\{\phi(x-j)\}_{j\in\mathbb{Z}}$ is a Riesz basis of V_0.

Statement 6 is slightly more obscure. It is motivated by the requirement that each of the approximation spaces should have a basis generated by the *scaling function*, $\phi(x)$, since the *wavelet*, $\psi(x)$, is constructed from the scaling function.

In summary, the scaling function, $\phi(x)$, may be used to generate a set of basis functions for each approximation space V_i. Similarly, the *wavelet* may be used to generate a set of basis functions for the detail spaces, W_i. Thus wavelet bases are appropriate for representing the detail required to move from one resolution to the next. Figure 3.2 gives an example of a scaling function and wavelet.

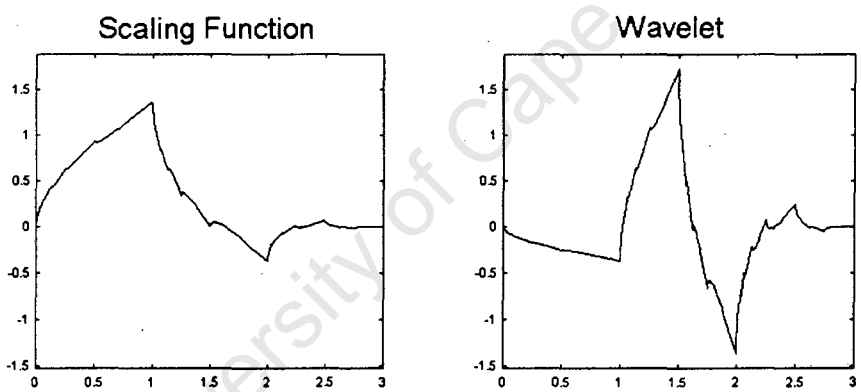


Figure 3.2: The Daubechies 4-coefficient scaling function, $\phi(x)$, and wavelet, $\psi(x)$ [1].

3.3 Filter Banks

The wavelet transform of a function is defined in terms of the inner product of that function and wavelet basis functions in $L^2(\mathbb{R})$, as shown in Equation 3.1. However, it can be shown that a relatively simple and efficient implementation of the wavelet transform may be achieved using linear filter banks [33]. This method is often referred to as the *pyramid algorithm* and uses low pass and high pass filters to obtain the *approximation* and *detail* coefficients, respectively.[7] (From an Electrical Engineering perspective, the concept of using low pass and

[7]Only the implementation *Discrete* Wavelet Transform will be discussed here.

high pass filters is perhaps more "natural".)

The following notation is commonly used for the filters that are used in the forward and inverse wavelet transforms:

- H_0: Low pass filter, forward wavelet transform.
- G_0: High pass filter, forward wavelet transform.
- H_1: Low pass filter, inverse wavelet transform.
- G_1: High pass filter, inverse wavelet transform.

It can be shown [33] that there is a direct relationship between the scaling function and the filter coefficients of H_0, and between the wavelet and the coefficients of G_0, as indicated in Equations 3.4 and 3.5.

$$\phi(x) = \sqrt{2}\sum_k h_k \phi(2x - k) \tag{3.4}$$

$$\psi(x) = \sqrt{2}\sum_k g_k \phi(2x - k) \tag{3.5}$$

where h_k and g_k are the coefficients of the filters H_0 and G_0 respectively.

3.3.1 The Pyramid Algorithm

The wavelet transform may then be implemented using the pyramid algorithm as follows: (The process is illustrated in Figure 3.3.)

- Convolve the signal with filter G_0 and down-sample by a factor of two to obtain the detail coefficients, $d_{i,j}$.[8]
- Convolve the signal with filter H_0 and down-sample by a factor of two to obtain the approximation coefficients, $c_{i,j}$.[9]
- Repeat the above process recursively, replacing the signal with the approximation coefficients.

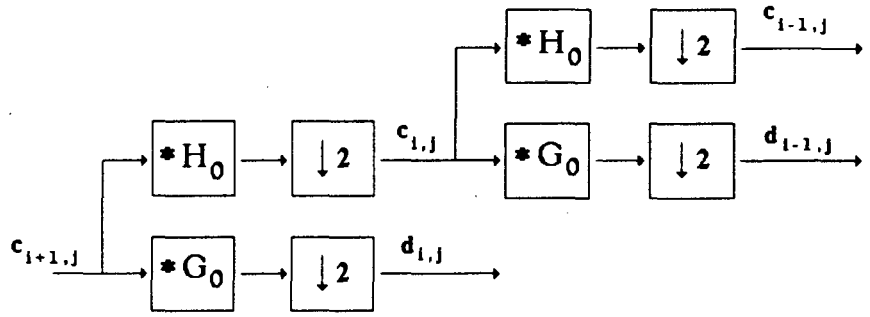


Figure 3.3: The forward wavelet transform implemented with convolution and down-sampling. Down-sampling is achieved by discarding every alternate value. (i indicates the resolution of the coefficients; j is a coefficient index.)

The inverse wavelet transform is simply the reverse of the procedures outline above. Figure 3.4 illustrates the process.

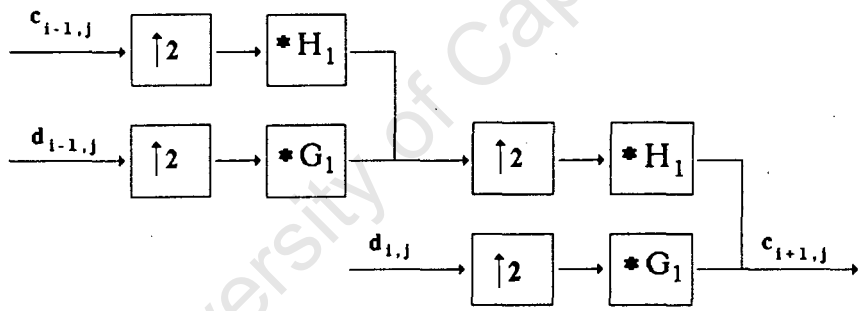


Figure 3.4: The inverse wavelet transform implemented with up-sampling and convolution. Up-sampling is achieved by inserting zeros between each pair of values.

An Example

Figure 3.5 illustrates an implementation of the wavelet transform using the pyramid algorithm. The signal is filtered and down-sampled to produce approximation and detail coefficients. The approximation coefficients are then recursively filtered and down-sampled to give approximation and detail coefficients at a lower resolution.

Following the decomposition process, it is possible to reconstruct the original signal through

[8] $d_{i,j} = \langle \psi_{i,j}, f \rangle$ are the detail coefficients at resolution i.
[9] $c_{i,j} = \langle \phi_{i,j}, f \rangle$ are the approximation coefficients as resolution i.

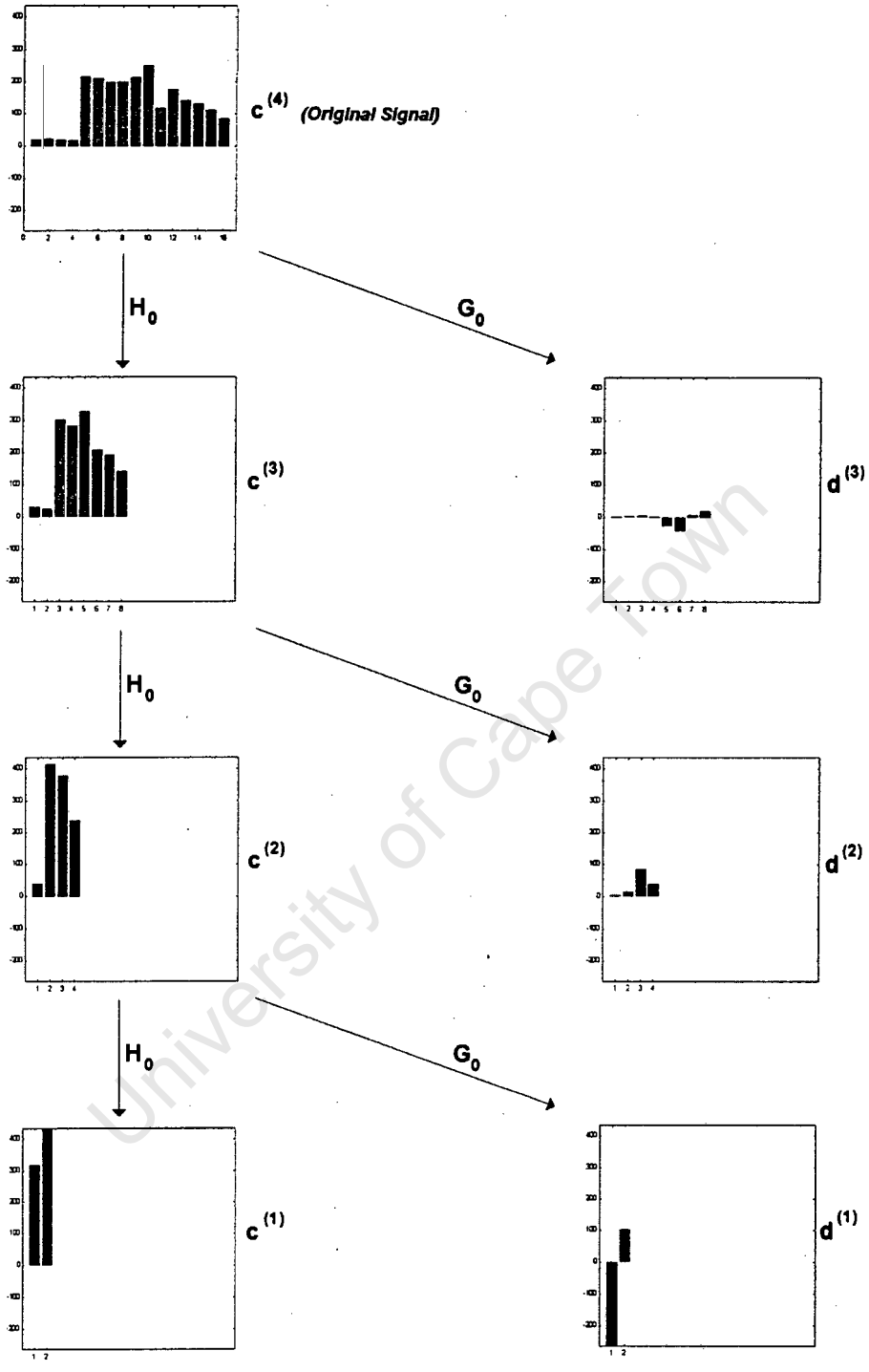


Figure 3.5: An example of *wavelet decomposition* (using the Haar filters [32]). The original signal consists of approximation coefficients in subspace V_4. The approximation coefficients are recursively filtered and down-sampled to obtain detail coefficients in subspaces W_3, W_2 and W_1, and approximation coefficients in subspace V_1. (In this example, a three-level decomposition is used.)

the inverse process of up-sampling and filtering. To do this, one needs the *lowest resolution* approximation coefficients and *all* the detail coefficients.

Alternatively, one may wish to reconstruct a coarse approximation of the original signal at a particular scale. This can done using the *lowest resolution* approximation coefficients and enough *detail coefficients* to give the desired resolution. This is illustrated in Figure 3.6. This figure also shows the manner in which approximation and detail coefficients are commonly arranged: first the lowest resolution approximation coefficients, and then the detail coefficients (in order of increasing resolution).

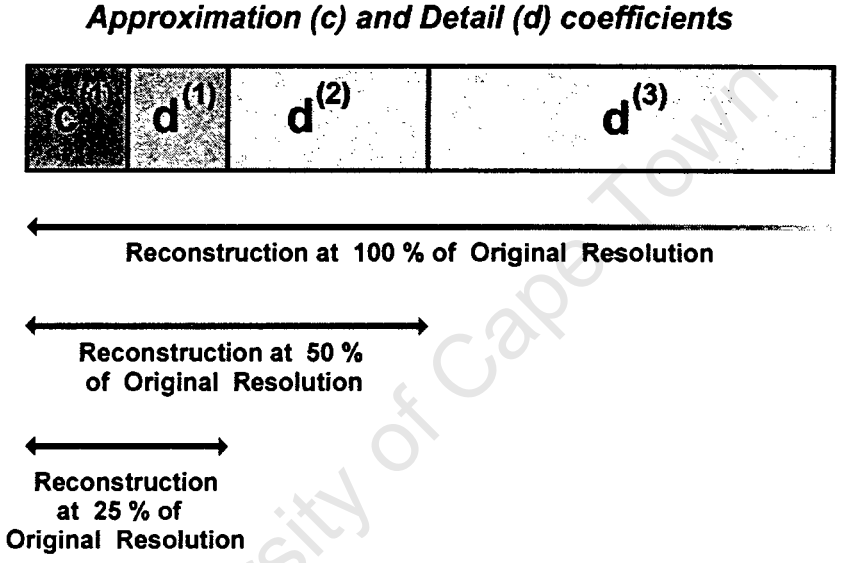


Figure 3.6: An example of signal reconstruction at a desired resolution using approximation and (selected) detail coefficients.

3.3.2 Boundary Extension

When a signal with N_S samples is convolved with a filter of length N_F, the output waveform has $N_S + N_F - 1$ samples. As a result, the number of approximation and detail coefficients is greater than the number of signal samples. From a compression perspective, increasing the number of coefficients is undesirable. However, several techniques are commonly used to eliminate the need for extra coefficients, without resulting in a loss of information. Two of these are discussed below.

Periodic extension is a technique used in computing the Fast Fourier Transform of a signal. It can also be applied to the wavelet transform. Periodic extension entails extending the signal

periodically at its boundaries, as in the example below.

Periodic Extension: $[1,2,3,4] \rightarrow [1,2,3,4,\ 1,2,3,4,\ 1,2,3,4]$

The disadvantage of periodic extension is that it may introduce discontinuities at the signal boundaries. A popular alternative is to use *symmetric extension*, which generally allows for a smoother signal at the boundary (as in the example below). The implementation of symmetric extension is slightly more complex than that of periodic extension because the form of extension (even or odd) may vary. A detailed description of symmetric extension is provided by Brislawn [3]. Note that symmetric extension requires the use of symmetric or anti-symmetric filters.

Symmetric Extension: $[1,2,3,4] \rightarrow [4,3,2,1,\ 1,2,3,4,\ 3,2,1]$

3.4 Achieving Compression Using the Wavelet Transform

When a signal is decomposed using the wavelet transform, the number of resulting approximation and detail coefficients is the same as the number of samples in the original signal.[10] Thus an application of the wavelet transform does *not* directly achieve compression of a signal. This section provides a brief overview of the role of the wavelet transform in compression.

Many "natural" signals (e.g. a sample of sound) have a high correlation between neighbouring samples. Thus a sample in such a signal will have a value which is (generally) close to those of its neighbours. Most "natural" images also contain a high degree of correlation between neighbouring pixels. Pixels in regions of an image (corresponding to objects within the image) generally have similar intensities. However, in images there are also edges at object boundaries, at which point there is little or no correlation between neighbouring pixels.

The intensity of a pixel in an image *sequence* is also generally well-correlated with the intensity of pixels in the *same* position which occur in subsequent or preceding frames. Thus a picture which forms part of a static background scene will have only marginal intensity changes over time. On the other hand, if there is a significant amount of motion in the vicinity of the pixel, or a scene change, this will result in a sharp change in pixel intensity.

The wavelet transform is well-suited to decorrelating such signals. Large regions (or periods) of similar pixel intensity can be expressed fairly accurately with only a few approximation coefficients. In such regions (or periods), the detail coefficients are likely to be largely insignificant. Edges (or transitions) are generally localised in space (or time) so that they may be described by detail coefficients at various scales.

[10] This assumes the use of periodic or symmetric extension.

Many wavelet-based codecs [28, 25, 12] typically exploit the above properties in order to compress signals with regions of high correlation [28]. Thus a signal is compressed by the efficient *coding* of a transformed signal. The coding process typically achieves compression by encoding large clusters of *insignificant detail coefficients* using only a few bits.

These two processes (the *transform* stage and the *coding* stage) are discussed in the next two chapters. Chapter 4 describes the application of the wavelet transform to digital video. Chapter 5 discusses the coding of the video once it has been transformed.

Chapter 4

The Transform Process

Chapter 3 introduced the wavelet transform and described how it may be used for decorrelating signals. A video can be represented as a signal in three dimensions (with two spatial dimensions and time dimension). Separable implementations of the wavelet transform exist, and thus the wavelet transform may be easily applied in multiple dimensions. This chapter provides a brief overview of the 2D wavelet transform (as commonly used in image compression) and discusses the use of the 3D wavelet transform as a method for transforming an image sequence.

4.1 The 2D Wavelet Transform

One of the most common applications of the wavelet transform in two dimensions is the compression of digital images. The two most popular methods of implementing the transform in two dimensions are both based on the one-dimensional wavelet transform.

The (n-level) *standard* decomposition is simply obtained by applying the (n-level) 1D transform separably to all the rows and then to the columns of an image.[1] The (n-level) *non-standard* decomposition is achieved by applying the (1-level) 1D transform to the rows and then to the columns of an image. This process is applied recursively ($n - 1$ times) to the remaining approximation coefficients.[1]

In most *image* compression codecs, the *non-standard* 2D wavelet transform is used. The main advantage of the non-standard decomposition (over the standard decomposition) is that it allows for a simple "parent-child" relationship between approximation and detail coefficients at the same position, over various scales (as illustrated in Figure 4.1) [28]. Related coefficients are grouped together in *Spatial Orientation Trees* [25].

[1] The order of rows or columns first is not important.

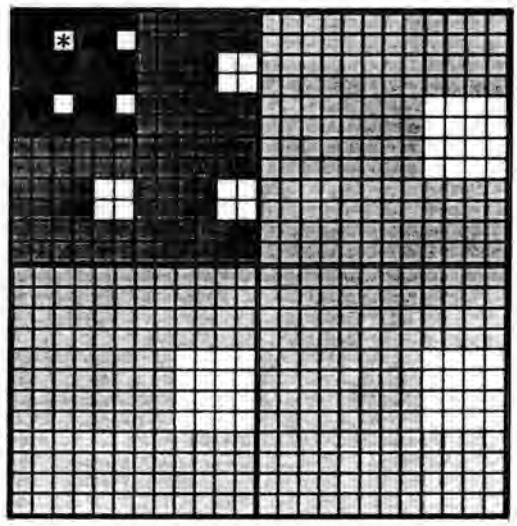


Figure 4.1: An example of a (three-level) 2D sub-band decomposition using the *non-standard* wavelet transform. The low resolution coefficients are coloured dark grey, while the higher resolution coefficients are shaded progressively lighter. The blocks in white illustrate a *spatial orientation tree*, $\mathcal{T}(i,j)$. The root of the tree (the coefficient $(i,j) = (2,3)$) is indicated by the approximation coefficient marked with a *. All the descendants of this coefficient (at various scales) are represented by the blocks coloured white.

4.2 The 3D Wavelet Transform

The 2D wavelet transform is widely used in *image* compression applications. It would thus seem reasonable to consider the 3D wavelet transform for video, with the third dimension being time. In this way, a three-dimensional group of frames (GOF) comprising part of an image sequence may be transformed to give a GOF of approximation and detail coefficients.

The number of frames in a GOF may be critical in various applications. When a GOF is transformed, it is necessary to buffer each of the frames in memory. Thus for a group comprising 16 frames, a delay of 16 frames would occur even before the transform is performed. In some video applications (such as video-conferencing), this would result in an unacceptable delay. For example, at a frame rate of 10 fps, a group comprising 16 frames would produce a delay of 1.6 seconds. Figure 4.2 illustrates the frames in a four-frame GOF.

A non-standard decomposition for three dimensions (similar to that for two) is possible. However, this would require that the same number of levels of decomposition be used in each of the three dimensions. In the case of a four-frame GOF, this would only allow for two levels of decomposition. Typically, the greater the number of levels of decomposition, the greater the degree of compression which can be achieved.

(a) Frame 1

(b) Frame 2

(c) Frame 3

(d) Frame 4

Figure 4.2: A group of four frames from the "Mother & Daughter" sequence. Note the slight rotation of the woman's head and the movement of her mouth.

Consequently, it was decided to use a 3D wavelet transform which consists of a 1D transform in the temporal direction and a non-standard 2D transform applied spatially. Figures 4.3 and 4.4 depict such a decomposition applied to the GOF in Figure 4.2. In Figures 4.3 and 4.4, the magnitudes of approximation and detail coefficients are represented by intensity. Thus the more significant coefficients are coloured brightly in these figures.[2]

(a) Frame 1

(b) Frame 2

(c) Frame 3

(d) Frame 4

Figure 4.3: The above four frames show the GOF in Figure 4.2 after having been transformed *temporally* using the (2-level) 1D wavelet transform. Frame 1 conveys the (temporal) DC information in the group. Note that the woman's head in Frame 1 appears somewhat blurred. This is because her head moves slightly over the four frame interval.

[2]Each coefficient, $W(i,j)$, has been scaled (for display purposes) using the intensity mapping $I(i,j) = log(1+|W(i,j)|)$.

(a) Frame 1 (W_{DC})

(b) Frame 2

(c) Frame 3

(d) Frame 4

Figure 4.4: The above four frames show the GOF in Figure 4.3 after having been further transformed *spatially* using the (4-level) 2D non-standard wavelet transform.

4.3 Inter-GOF Prediction

The role of the wavelet transform in image or video compression is to describe an image or a GOF at various scales, so that this information may be more easily compressed. However, each GOF (as described above) is a distinct entity. This section examines correlations between successive GOFs and discusses methods for eliminating redundant information, so as to further improve compression.

4.3.1 DC Prediction

Generally, there is relatively little motion from one frame to the next in a typical video sequence. Thus one would expect the *DC* coefficients in a temporally-transformed GOF to convey a significant amount of information. (The remaining frames in the transformed GOF convey motion information.) The DC coefficients are the approximation coefficients in the temporal dimension. Therefore the DC frame is the weighted mean of all the frames in the GOF, multiplied by a constant.

Some areas of a scene (e.g. the background) may be stationary over several GOFs. Under such conditions, the DC frame of one GOF would be fairly similar to the DC frame of the previous GOF. Thus an alternative to coding the DC frame of each GOF would be to encode the *difference* between consecutive DC frames. The difference frame would generally have a substantially lower entropy.[3]

Figure 4.5(a) shows an example of a *DC difference* frame (ΔW_{DC}). It is evident that this frame contains fewer significant coefficients than Frame 1 in Figure 4.4 (W_{DC}).

4.3.2 Selective DC Prediction

In the case of a scene change between GOFs or even a significant amount of motion in part of a scene, the DC difference would generally be substantial. In this case it would be preferable to encode the actual DC frame, as opposed to the difference from the previous one.

Consequently, a method was devised to allow a combination of the above two techniques, depending on the nature of the scene. The method is based on considering *regions* in the DC frame, and determining whether to predict the DC coefficients in that region. However, each "region" is defined as a spatial orientation tree in the (2D) *wavelet* domain of the DC frame. As a result of this, these regions do not have sharp boundaries in the spatial domain.

The algorithm for the selective prediction of DC coefficients (including the definition of "re-

[3]For an overview of *entropy*, refer to Section 5.4.

(a) ΔW_{DC} (b)

Figure 4.5: A DC *difference* frame (ΔW_{DC}), representing the difference between the DC frames of the current and previous GOFs. (a) The DC difference frame (in the 3D wavelet domain); (b) The same DC difference frame depicted in the (temporal) 1D wavelet domain (for display purposes).

gions") is presented below. First, some terminology is introduced:[4]

Terminology

- Assume that a group of image frames has been transformed *spatially and temporally* to produce a 3D GOF of wavelet coefficients. Let the *DC frame* be denoted by W_{DC}, and the *difference* between W_{DC} and the *previous* DC frame by ΔW_{DC}.
- Let $\mathcal{T}(i,j)$ represent the family of coefficient (i,j), namely the set comprising (i,j) and *all* its descendents in the *spatial orientation tree*. Refer to Figure 4.1 for an example.
- Let $S(X)$ denote the *entropy* of the set of coefficients, X.
- Let $\mathcal{H}$ denote the set of all spatial orientation tree *roots* in a frame - i.e. the (spatial) approximation coefficients. Then, referring to Figure 4.1 as an example, $\mathcal{H}$ would consist of the nine approximation coefficients (which are depicted in the top left corner of the coefficient grid).
- In addition, let $\mathcal{L}(i,j)$ denote the set of all spatial orientation tree nodes which are the *descendants of the the children* of the node (i,j). Then, referring to Figure 4.1 as an

[4]Note that some of the terms introduced here are similar to those used in SPIHT. Refer to Section 5.2 and [25].

example, $\mathcal{L}(2,3)$ would consist of all the coefficents shaded white, *except for* (i) the root node, $(2,3)$ and (ii) its three children.

- Finally, let $\mathcal{M}$ denote a "prediction map". $\mathcal{M}$ is a binary matrix of the same dimensions as $\mathcal{H}$. A value of 0 for $\mathcal{M}(i,j)$ indicates that prediction is used for for the spatial orientation tree, $\mathcal{T}(i,j)$. Conversely, a value of 1 for $\mathcal{M}(i,j)$ shows that no prediction is used for for $\mathcal{T}(i,j)$.

The Selective Prediction Algorithm

Given the *DC frame*, W_{DC}, and the *DC difference frame*, ΔW_{DC}, (which effectively convey identical information[5]) the purpose of the *Selective Prediction of DC* algorithm is to produce a *modified DC* frame, $\widetilde{W}_{DC}$, which has lower entropy than either W_{DC} or ΔW_{DC}.

Let $\widetilde{W}_{DC} = \Delta W_{DC}$
Let $\mathcal{M} = \mathbf{0}$
for each $(i,j) \in \mathcal{H}$,
  $S_{W_{DC}} = S\left(\{W_{DC}(p,q)\}_{(p,q)\in\mathcal{T}(i,j)}\right)$
  $S_{\Delta W_{DC}} = S\left(\{\Delta W_{DC}(p,q)\}_{(p,q)\in\mathcal{T}(i,j)}\right)$
  if $S_{W_{DC}} < S_{\Delta W_{DC}}$ then
    for each $(p,q) \in \mathcal{T}(i,j)$
      $\widetilde{W}_{DC}(p,q) = W_{DC}(p,q)$
    end
    $\mathcal{M}(i,j) = 1$
  end
end

Using the above algorithm, the combination of the modified DC frame, $\widetilde{W}_{DC}$, and the prediction map, $\mathcal{M}$, convey the same information as the original DC frame, W_{DC}.[5]

An Alternative to Entropy

In practice, calculating the entropy of a spatial orientation tree may be computationally expensive. Consequently, it was decided to use an alternative measure, based on the absolute value of the coefficients in the tree. Thus, for $(i,j) \in \mathcal{H}$, $S_{W_{DC}}(i,j)$ is instead calculated as follows:[6]

[5] This assumes that the DC frame from the previous GOF is known.
[6] A similar method is used for calculating $S_{\Delta W_{DC}}(i,j)$

$$S_{W_{DC}}(i,j) = \sum_{k} |X_k|_{(X_k \in X)} \tag{4.1}$$

where

$$X = \{W_{DC}(p,q)\}_{(p,q)\in\mathcal{L}(i,j)}$$

The above measure is based on the hypothesis that the tree which has more coefficients clustered around zero will have a lower entropy. It is important to stress that $S(i,j)$ has *not* been defined as a general "approximation" of entropy, but rather as an *alternative* measure of the extent to which (detail) coefficients are clustered around zero.

Figure 4.6(a) illustrates an example of a selectively-predicted DC frame for which the alternative entropy measure of Equation 4.1 was used to decide on prediction for each region. It is evident that this frame contains fewer significant coefficients than Frame 1 in Figure 4.5 (ΔW_{DC}). Figure 4.6(b) shows that the background and the girl (who moved only slightly) were predicted from the previous DC frame. In contrast, the region around the woman (who moved more substantially) was *not* predicted.

4.3.3 A Comparison of DC Prediction Methods

Thus far, two DC prediction techniques have been presented as a means of achieving entropy reduction, and hence improving compression performance. This section illustrates (by means of an example) the *distribution* of coefficients obtained when one of the above prediction methods is applied to a DC frame within a transformed GOF. Clearly, the effectiveness of each of the above methods will depend largely on the nature of the video sequence and the amount of motion.

Figure 4.7 shows a DC frame[7] with (a) no prediction, (c) prediction, and (e) selective prediction. Alongside each DC frame is a histogram showing the distribution of the magnitude of the coefficients in that frame. From the histograms, it is evident that the coefficients in each of the frames have magnitudes which are clustered around zero. The histogram of the selectively predicted frame ($\widetilde{W}_{DC}$) shows that *almost half* of the coefficients in this frame have a value of zero - a greater proportion than for either of the other two frames.

The advantage of a distribution characterised by a few values occurring with high probability

[7]This DC frame was obtained by transforming the GOF in Figure 4.2.

(a) $\widetilde{W}_{DC}$

(b)

(c) M

Figure 4.6: (a) A selectively-predicted DC frame ($\widetilde{W}_{DC}$) depicted in the 3D wavelet domain; (b) The same frame depicted in the (temporal) 1D wavelet domain (for display purposes); (c) The binary *prediction map* that indicates which spatial orientation trees in the DC frame have been predicted.

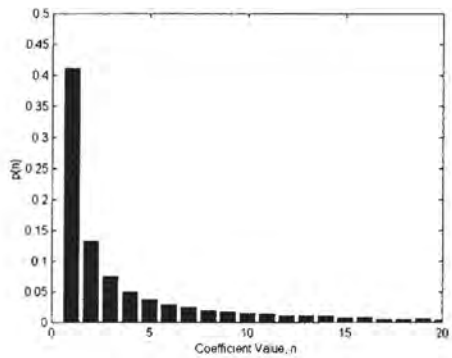
is that frequently-occurring symbols can be coded very efficiently. This translates into a greater degree of compression. Table 4.1 gives the entropy (or average information) of each of the three frames in figure 4.7. In addition to entropy, the table indicates the value of

$$\mathcal{S} = \sum_{(i,j)\in\mathcal{H}} S(i,j)$$

for each frame. Recall that $S(i,j)$ was used as an alternative measure to entropy. (Refer to Equation 4.1.)

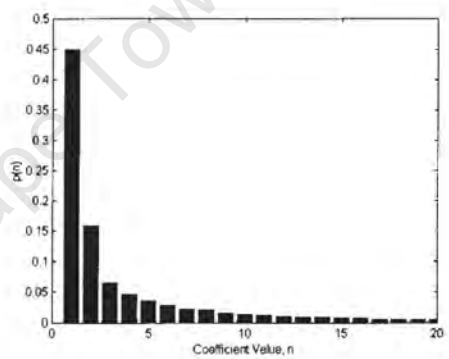
It is evident that the selectively-predicted DC frame has the lowest entropy (and the lowest value of $\mathcal{S}$) out of the three frames. The advantage of *Selective DC Prediction* is that even in the *worst* case,

(a) W_{DC}



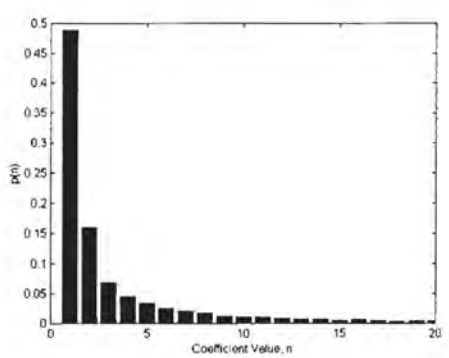
(b) Histogram of W_{DC}

(c) ΔW_{DC}



(d) Histogram of ΔW_{DC}

(e) $\widetilde{W}_{DC}$



(f) Histogram of $\widetilde{W}_{DC}$

Figure 4.7: An illustration of the distribution of the magnitude of approximation and detail coefficients in a DC frame. (a) and (b): No Prediction. (c) and (d): Prediction. (e) and (f): Selective Prediction.

$$\mathcal{S}_{\widetilde{W}_{DC}} = min\{\mathcal{S}_{W_{DC}}, \mathcal{S}_{\Delta W_{DC}}\}$$

The only overhead associated with selective prediction is that a binary prediction map, $\mathcal{M}$, is required for specifying which spatial orientation trees represent *difference* information.

Table 4.1: An example of entropy reduction using selective prediction.

DC Frame	Entropy	$\mathcal{S}(\times 10^5)$
W_{DC}	3.79	1.96
ΔW_{DC}	3.36	1.44
$\widetilde{W}_{DC}$	3.08	1.12

4.4 Extension to Colour

In the preceding discussion, only the *luminance* (Y) component of a video sequence has been considered. In colour sequences there are also two chrominance components (e.g. C_b and C_r). Separate GOFs can be formed for each of the chrominance components, and chrominance GOFs can likewise be transformed using the 3D wavelet transform.

Selective Prediction of DC frames can also be used independently for each of the chrominance components. However, it would be reasonable to expect that *motion* in the chrominance bands is fairly well correlated with *motion* in the luminance component. Assuming this to be true, one could use the luminance prediction map, $\mathcal{M}$, for the chrominance components too.

The chrominance components of a video signal are frequently sub-sampled spatially to half the resolution of the luminance component.[8] In this case it *is* still possible to use only the luminance prediction map, $\mathcal{M}$, provided that (spatially) the number of levels of the 2D wavelet transform of the chrominance components is *one fewer* than the number of levels of the 2D wavelet transform of the luminance component. This would ensure that the size of $\mathcal{H}$ is the same for the luminance and each of the (sub-sampled) chrominance components.

[8]This is referred to as 4:2:0 spatial sub-sampling.

Chapter 5

The Coding Process

As outlined in Section 3.4, the application of the wavelet transform to a signal does not automatically result in compression. It is the *coding* stage of a codec which allows for transformed coefficients to be described efficiently, thus achieving compression. This chapter first provides an overview of some successful image and video coding techniques which have been applied to wavelet-transformed images and sequences. Following this, an alternative coding scheme for video is presented.

5.1 Progressive Transmission

Progressive Transmission is a feature common to several wavelet-based image codecs, including Shapiro's method of *Embedded Image Coding Using Zerotrees of Wavelet Coefficients* (EZW) [28] and Said and Pearlman's technique of *Set Partitioning in Hierarchical Trees* (SPIHT) [25]. Similar video codecs have also been developed which allow for progressive transmission [31, 5, 11]. This section provides a theoretical overview of progressive transmission and discusses its relevance for the transmission of compressed video bit-streams.

5.1.1 Theory of Progressive Transmission

Consider the one-dimensional digital signal defined by the vector $\mathbf{s}$.[1] Let $\mathbf{s}$ be transformed using a *unitary* transform such as the wavelet decomposition (represented by $\mathcal{W}$) to produce the sequence of approximation and detail coefficients, $\mathbf{c}$.

[1] Much of the following explanation of Progressive Transmission is based on that given by Said and Pearlman [25] for the case of a (2D) image.

$$\mathbf{c} = \mathcal{W}(\mathbf{s})$$

Assume that a coding algorithm is used to encode the coefficients of $\mathbf{c}$ as a progressively transmitted bit-stream. Then, starting with an initial approximation of $\hat{\mathbf{c}} = \mathbf{0}$, the decoder will decode the bits from the bit-stream as they are received in order to refine the approximation, $\hat{\mathbf{c}}$, of the coefficients, $\mathbf{c}$. An approximation, $\hat{\mathbf{s}}$, of the original signal, $\mathbf{s}$, may then be obtained using the inverse transform, $\mathcal{W}^{-1}$.

$$\hat{\mathbf{s}} = \mathcal{W}^{-1}(\hat{\mathbf{c}})$$

The essence of progressive transmission is to encode the most important information first. Various measures may be used for determining the importance of information. A common approach is to specify the most important information as that which allows for the greatest *reduction in error* between the original and reconstructed signals. Assuming a *mean square* measure, the error in reconstruction, ε is given by:

$$\begin{aligned} \varepsilon(\mathbf{s} - \hat{\mathbf{s}}) &= \frac{1}{N}|\mathbf{s} - \hat{\mathbf{s}}|^2 \\ &= \frac{1}{N}\sum_i (s_i - \hat{s}_i)^2 \end{aligned} \tag{5.1}$$

where N is the number of samples in $\mathbf{s}$. Recall that the transform, $\mathcal{W}$, is *unitary*.[2] It therefore preserves Euclidean distance, so that:

$$\begin{aligned} \varepsilon(\mathbf{s} - \hat{\mathbf{s}}) &= \frac{1}{N}|\mathbf{c} - \hat{\mathbf{c}}|^2 \\ &= \frac{1}{N}\sum_i (c_i - \hat{c}_i)^2 \end{aligned} \tag{5.2}$$

Equation 5.2 implies that each coefficient, c_i, which is decoded (at the decoder) reduces the *mean square error* (MSE) of the reconstructed signal, $\mathbf{s}$, by an amount equal to $\frac{1}{N}c_i^2$. (Recall that $\hat{\mathbf{c}}$ was initialised to $\mathbf{0}$.) Consequently, progressive transmission can be performed by

[2]This is not true in the case of a *biorthogonal* wavelet transform. However, many biorthogonal transforms are *approximately* unitary. [30, p. 70]

coding coefficients in order of the square of their value, with the largest-valued coefficients being coded first. Note that this is equivalent to coding coefficients in order of their *magnitude*.

If c_i is represented as a binary number (of finite precision), then it is evident that the most significant bit of c_i conveys more information (i.e. contributes more to the reduction in MSE) than any of the other bits in c_i. This concept is encapsulated in the *bit-plane* progressive transmission method of [22] which allows for the more significant bits of a coefficient to be coded first.

Progressive Transmission of Difference Information

In the previous chapter, the concept of *DC prediction* was discussed. This entails the prediction of a coefficient (by approximating it with its previous value in time). The *difference* value is then encoded. This section briefly examines the information conveyed by *difference coefficients* within a progressive transmission context.

Referring to section 5.1.1 (and particularly Equation 5.2), it is evident that the only change when prediction is used, is that $\hat{\mathbf{c}}$ is not *initialised* to $\mathbf{0}$, but to the value from which it is predicted (i.e. its previous value). Thus

$$\hat{\mathbf{c}}_{t_i}(\text{initial value}) = \hat{\mathbf{c}}_{t_{i-1}}$$

where t_i represents the current coding process and t_{i-1} represents the previous coding process. Consequently, *selective prediction* may easily be integrated into a progressive transmission scheme, provided that both the encoder and decoder have prior information as to which coefficients are to be predicted.[3]

5.1.2 A Generic Implementation of Progressive Transmission

Wavelet-based image codecs such as EZW and SPIHT use progressive transmission for encoding approximation and detail coefficients. This section describes a generic implementation of progressive transmission, which has features common to both EZW and SPIHT.[4]

Consider a one dimensional sequence of non-negative, real-valued coefficients, $\{c_i\}_{i\in\mathbb{Z}}$.[5] Let

[3] This could be provided using a (binary) prediction map, for instance.

[4] The discussion presented here is based on a similar generic method for progressive transmission, described by Said and Pearlman [25].

[5] The algorithm presented here can easily be extended to handle real-valued coefficients in multiple dimensions.

these coefficients be rearranged according to the minimum number of bits required to represent the value of each coefficient. Define $\eta : I \rightarrow I$ as a one-to-one mapping which specifies the new *ordering* of the coefficients, satisfying the inequality

$$\lfloor \log_2 c_{\eta(k)} \rfloor \geq \lfloor \log_2 c_{\eta(k+1)} \rfloor \tag{5.3}$$

Thus the original set of N coefficients, $\{c_1, c_2, c_3, \ldots, c_N\}$, is rearranged according to Equation 5.3 to give the same coefficients, in a different order: $\{c_{\eta(1)}, c_{\eta(2)}, c_{\eta(3)}, \ldots, c_{\eta(N)}\}$.

Table 5.1 illustrates an example with $N = 6$ coefficients. The binary representation of each coefficient is also shown. All of the bits in a column convey information of equal significance, with the most significant bits on the left. Table 5.2 shows the same set of six coefficients sorted according the number of bits required for their binary representation.

Table 5.1: Six coefficients and their binary representation

Coefficient	Decimal Value	Binary Value
c_1	5	0 1 0 1
c_2	3	0 0 1 1
c_3	2	0 0 1 0
c_4	12	1 1 0 0
c_5	9	1 0 0 1
c_6	3	0 0 1 1

Table 5.2: The six coefficients from Table 5.1 sorted according to Equation 5.3

k	$\eta(k)$	Coefficient	$c_{\eta(k)}$
1	4	c_4	**1 1 0 0**
2	5	c_5	**1 0 0 1**
3	1	c_1	**1 0 1**
4	2	c_2	**1 1**
5	3	c_3	**1 0**
6	6	c_6	**1 1**

Define μ_n as the *number* of coefficients for which $2^n \leq c_i < 2^{n+1}$. For the example in Table 5.2: $\mu_3 = 2$, $\mu_2 = 1$ and $\mu_1 = 3$.

Once the coefficient ordering information and the values of μ_n have been specified, the coefficients can be encoded using a progressive transmission technique. This can be done by

sequentially encoding the bits in each column, as illustrated by the bits in boldface in Table 5.2. Note that because the coefficients are arranged in decreasing order of magnitude and because μ_n has been specified, the leading "0" bits and the first "1" of each row need not be specified. The details of the algorithm are outlined below.

Algorithm for Progressive Transmission

1. Output $n = \lfloor \log_2(max\{c_i\}) \rfloor$
2. **Sorting Pass:** Output μ_n, and $\eta(k)$ for each of the μ_n coefficients satisfying $2^n \leq c_{\eta(k)} < 2^{n+1}$.
3. **Refinement Pass:** Output the n^{th} most significant bit of all coefficients satisfying[6] $c_i \geq 2^{n+1}$, in the same order in which the $\eta(k)$ were specified during the sorting pass.
4. Decrement n and go to Step 2.

The coding proceeds until either the desired number of bits or degree of approximation has been achieved. The algorithm for the *decoding* process is the same as that for encoding, except that "Output" is replaced by "Input".

5.1.3 Progressive Transmission of Video Data

Consider a bit-stream generated using a *progressive transmission* technique. The initial portion of such a bit-stream conveys the most important information (e.g. a low resolution approximation), while the latter portion of the bit-stream conveys less significant data (e.g. the finer details both temporally and spatially). This has very important implications for the transmission of video over a channel, since portions of the bit-stream can then *easily* be assigned a *priority* depending on their location in the bit-stream for a GOF.

Assume, for example, that a video sequence is encoded using a progressive transmission method and transmitted over an Asynchronous Transfer Mode (ATM) network at a rate of 1.5 Mbps. It may be that the network is congested and is only able to deliver the bit-stream to the decoder at a rate of 1.0 Mbps, say. In this case, the network could simply deliver those cells with high priority and discard some of those with low priority. The decoder would then be able to decode the initial portion of the bit-stream for each GOF and reconstruct the video sequence at a reduced (but reasonable) quality. This example is illustrated in Figure 5.1.

[6]Those coefficients already specified during a *previous* sorting pass.

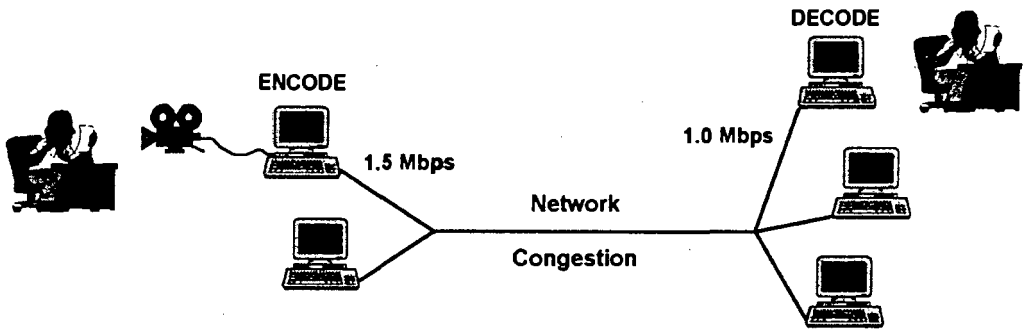


Figure 5.1: An example of the *progressive transmission* of a compressed video bit-stream over a congested network. Despite channel congestion, the network can still deliver the more significant portions of the bit-stream - which will allow for a reasonable quality sequence to be decoded.

Progressive transmission is also particularly advantageous for the implementation of multi-casting, multiresolution encoding and joint source-channel coding. More details can be found in [23].

5.2 SPIHT for Image Coding

Set Partitioning in Hierarchical Trees (SPIHT) [25] is a *state-of-the-art* wavelet-based image coding technique that was developed by Said and Pearlman in 1996. SPIHT is closely related to EZW, but has exhibited superior performance and is significantly less reliant on entropy coding [25]. An outline of the SPIHT algorithm is provided below.

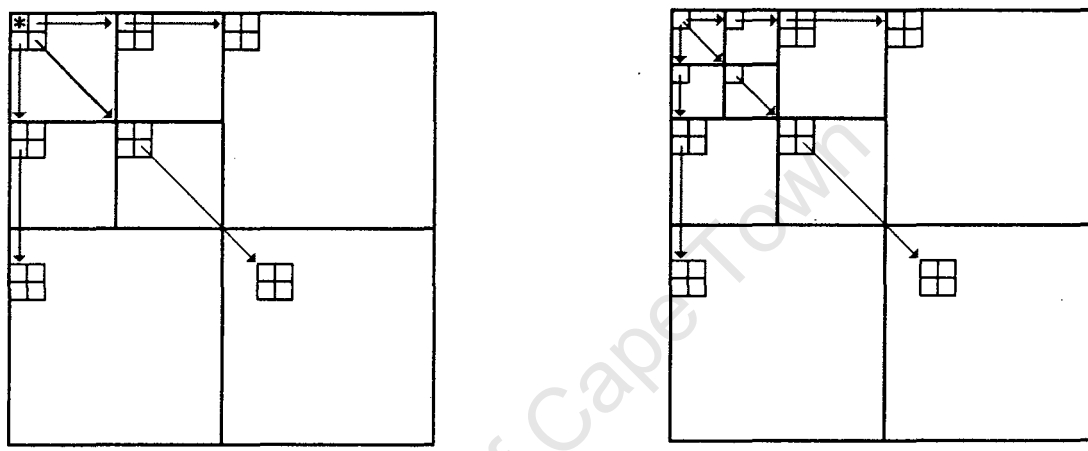
Given a wavelet-transformed image, SPIHT attempts to code the coefficients as efficiently as possible. It does this by stepping through *sorting* and *refinement* stages until a desired stopping condition is reached.[7]

During the *sorting* stage, the significance (or insignificance) of pixels or groups of pixels with respect to a threshold[8] is determined. Coefficients are grouped in *spatial orientation trees*, so that they convey information about the same region of an image at various scales. In the EZW coding method, the significance of coefficients and trees of coefficients is specified by scanning through the coefficients in a predefined order. However, the SPIHT algorithm uses an *adaptive* scanning order which is determined by those coefficients previously coded as significant.

[7]Typically, until the compressed data stream has reached a specified size.
[8]Each threshold is a power of two.

During the *refinement* stage of SPIHT, those coefficients previously encoded as *significant* are specified with an additional level of accuracy using a bit-plane method.

It is a requirement of SPIHT that all root nodes (spatial approximation coefficients) be arranged in groups of 2×2, made up of adjacent coefficients. One node in each group of four is defined to have *no* descendants (as depicted by the asterisk in Figure 5.2). The reason for such a 2×2 grouping of coefficients is that it was found to give an improved compression performance following the application of *arithmetic coding*, as described in [25].



(a) (b)

Figure 5.2: (a)Spatial orientation trees in SPIHT. (b) Spatial orientation trees in EZW and Modified SPIHT.

The SPIHT algorithm is implemented using three *lists* of coefficients: a list of insignificant pixels (LIP), a list of significant pixels (LSP) and a list of insignificant sets (LIS). The LIP and LSP contain the location of individual coefficients, while the LIS represents sets (trees) using the root node of each tree. Full implementation details are provided by Said and Pearlman [25].

5.3 SPIHT for Video Coding

This section discusses the use of SPIHT for the coding of digital video. First, an overview of 3D SPIHT [11, 12] (also co-developed by Pearlman) is presented. Following this, an alternative version of (2D) SPIHT is introduced as a video coding algorithm.

5.3.1 3D SPIHT

Following the success of SPIHT as an image codec, Kim, Xiong and Pearlman developed 3D SPIHT as a video codec [11, 12]. Just as (2D) SPIHT operates on a 2D wavelet-transformed image, so 3D SPIHT is intended for a *group* of frames (GOF) which has been decomposed using a 3D wavelet decomposition.

In 3D SPIHT, a three-dimensional *spatio-temporal orientation tree* is used. Once again, pixels are grouped in blocks, though these cubes are of dimension $2 \times 2 \times 2$. The core of the SPIHT algorithm involves specifying the significance of coefficients in the three lists (LIP, LSP and LIS). This is effectively dimension *independent*. Consequently, once a 3D tree structure has been defined, the implementation of 3D SPIHT is similar to that of its 2D counterpart.

For a reasonable number of temporal decomposition levels, a relatively large number of frames in a GOF is required. (For example, 3D-SPIHT requires 16 frames for a three-level decomposition, and 32 frames for a four-level decomposition.) As highlighted in Section 4.2, this introduces a delay which may be unacceptable in real-time implementations. The next section presents an alternative coding scheme to 3D SPIHT which does not require such a large GOF.

5.3.2 Modified 2D SPIHT

In this section, a modified implementation of SPIHT is presented as an algorithm for the coding of video. The modifications include an alternative *spatial* tree structure and a method for the interleaving of SPIHT bit-streams.

An Alternative Tree Structure

As its name suggests, SPIHT uses spatial orientation trees to group approximation and detail coefficients. In standard (2D) SPIHT, one in every four root nodes does not have any descendants. In addition, it is required that the number of approximation coefficients (root nodes) be arranged in groups of 2×2. This implies that $\mathcal{H}$ (as defined in Section 4.3.2) should have even dimensions (horizontally and vertically).

The following proposed simple modifications to SPIHT eliminate the above requirements: First, root nodes need not be arranged in 2×2 groups. Secondly, each root node (approximation coefficient) in a spatial orientation tree is defined to have exactly three children - the three detail coefficients in the same spatial location in the lowest-resolution detail space. An example of this tree structure is depicted in Figure 5.2(b).

It is worth noting that the above branching method for a spatial orientation tree is based on the spatial orientation trees used in Shapiro's EZW algorithm [28], to which SPIHT is closely

related. (Shapiro refers to *spatial orientation trees* as *wavelet trees*.)

The modified spatial orientation tree structure presented above enables SPIHT to handle *one additional layer* of (spatial) wavelet decomposition, since there is no longer any requirement to group root nodes in 2×2 blocks. Allowing for this additional layer of decomposition may help to obtain a greater degree of compression.

Application to Video Coding

As highlighted above, 3D SPIHT requires a relatively large GOF of size 16 to allow for three levels of temporal decomposition. For this reason, it was decided to not use three dimensional spatio-temporal trees. Instead, 2D SPIHT (with the modified version of tree structure described above) was extended to handle video.

The primary motivation for the use of modified 2D SPIHT for the coding of video is that the number of spatial and temporal decomposition levels applied to a GOF may differ. As will be shown, this is because (2D) spatial orientation trees are used. In contrast, the *spatio-temporal* orientation trees in 3D SPIHT typically require the same number of decomposition levels in the spatial and temporal dimensions. Consequently, modified 2D SPIHT may be used to code a GOF with relatively few (e.g. four) frames, without this limiting the number of levels of spatial decomposition. It is often desirable to limit the length of GOFs (depending on the video frame rate) so as to prevent delay in real-time applications.

The SPIHT algorithm describes the significance (or otherwise) of approximation and detail coefficients or groups of coefficients with respect to some threshold, 2^n. At the start of the SPIHT coding process, n is large, so that the more significant approximation and detail coefficients are coded (coarsely) first. As the coding progresses, the value of n is decremented regularly so that the less significant coefficients are also coded into the resultant bit-stream. In addition, those coefficients previously coded are described with an additional degree of refinement.

The above description is only a very brief summary of how SPIHT works, but it illustrates that for each value of n, the approximation and detail coefficients are coded with increasing accuracy. Thus, for each threshold, there is a contiguous stream of bits generated by the SPIHT algorithm. These are concatenated together (in order of decreasing n) to produce the final SPIHT bit-stream.

2D SPIHT can be used in the compression of *video* by coding each frame in a GOF separately, using the same threshold, and then *interleaving* the resultant bit-streams. The process is then repeated using a lower threshold each time. By interleaving the bit-streams corresponding to each frame, one is still able to ensure that for each threshold, there is a contiguous stream of

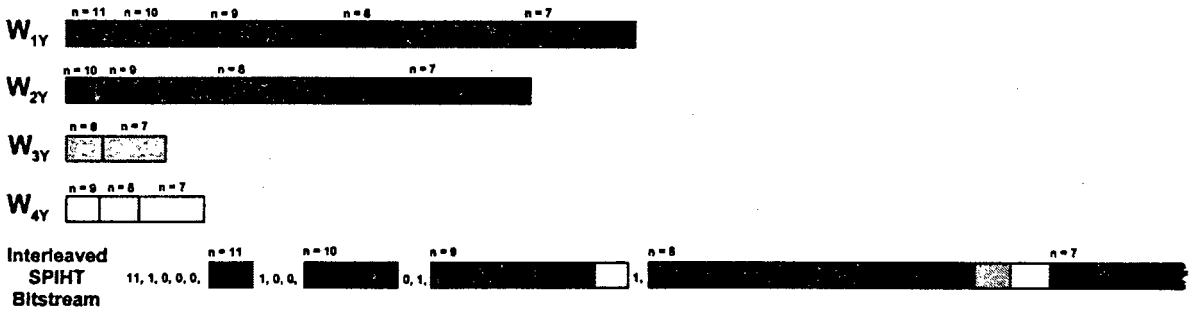


Figure 5.3: An Example of 2D SPIHT for Video, assuming a 4-frame luminance (Y) GOF: The SPIHT bit-streams for each frame are show, together with the corresponding threshold values, as represented by 2^n. The resultant interleaved SPIHT bit-stream is also shown.

bits generated by SPIHT.

In standard SPIHT (for image coding), the initial bits in the bit-stream are used to specify the starting value of n (and hence the initial threshold) [25]. When coding a GOF using SPIHT, each frame in the transformed GOF may have a different initial value for n. However, only the *maximum* initial value for n need be specified at the start of the bit-stream. The actual initial values of n for each frame may then be specified (relative to this maximum) using additional bits, as outlined in the example below.

Figure 5.3 illustrates an example of the interleaving process for a 4-frame luminance GOF. Note that in the case of two chrominance components, there would be a total of *twelve* bit-streams to interleave (four for each colour plane). However, the procedure would be similar to that outlined above.

An Example Given four SPIHT bit-streams (one for each frame of a 4-frame, luminance-only GOF), as depicted in Figure 5.3, an *interleaved* bit-stream may be generated as follows:

First, the maximum of the four initial values of n (i.e. eleven) is encoded. The next bit (1) indicates that the *first* bit-stream has $n_{initial} = 11$, while the following three bits (0,0,0) show that *none* of the other three bit-streams has $n_{initial} = 11$. Then follows that part of the bit-stream for frame 1 which corresponds to a threshold of 2^{11}.

The next bit (1) indicates that the *second* bit-stream has $n_{initial} = 10$, while the following two bits (0,0) show that *neither* of the other two bit-streams has $n_{initial} = 10$. Then follow those parts of the bit-streams for frames 1 and 2 which correspond to a threshold of 2^{10}.

The next bit (0) indicates that the *third* bit-stream does not have $n_{initial} = 9$, while the following bit (1) shows that the *fourth* bit-stream has $n_{initial} = 9$. Then follow those parts of the bit-streams for frames 1, 2 and 4 which correspond to a threshold of 2^9.

The next bit (1) indicates that the *third* bit-stream has $n_{initial} = 8$. (At this point, the values of $n_{initial}$ for *all* frames in the GOF are known.) Then follows those parts of the bit-streams for frames 1, 2, 3 and 4 which correspond to a threshold of 2^8. The interleaving of bit-streams continues in this order (W_{1Y},W_{2Y},W_{3Y},W_{4Y}) for each subsequent threshold, until the desired number of bits is reached.

5.4 Entropy Coding

Entropy coding is a *lossless* data compression technique. Given a set of symbols, the purpose of entropy coding is to encode the set efficiently in such a way that the more common symbols are represented by very short code words, while the less common symbols are represented by longer code words.

The *information* I, conveyed by a symbol s_i, is inversely proportional to the *probability*, p, of that symbol occurring. More precisely,

$$I(s_i) = -\log_2 (p(s_i)) \tag{5.4}$$

where I is measured in *bits*. This measure of information gives an (ideal) measure of how many bits are required to represent a specific symbol. The first order *entropy*, H, of an alphabet is the (weighted) average information per alphabet symbol. The contribution of each symbol to the entropy value is weighted by the probability with which that symbol occurs:

$$H = \sum_i p(s_i)\, I(s_i)$$

Two common forms of entropy coding are Huffman Coding [8] and Arithmetic Coding [13, 15]. In both of these methods, a symbol is represented by a code word which (ideally) has a length equal to the information conveyed by that symbol (as defined in Equation 5.4). Arithmetic coding generally demonstrates superior compression performance due to the fact that it allows code symbols to be of non-integer length. This is not the case with Huffman Coding.

From the above discussion, it is evident that the probability of a symbol (and hence the information conveyed by it) is determined by the distribution of symbols within a set. If this distribution is generally fixed over time, the entropy encoder and decoder can assume a fixed model of the data.

However, in most practical implementations, the distribution of symbols changes over time (depending on the data being coded). In such a case, it would be more effective for the encoder and decoder to use an *adaptive* distribution model as the data changes.

Chapter 6

System Overview: The Design of the *SPRED* Video Codec

The preceding two chapters have demonstrated how the wavelet transform may be applied to video, and how the resulting approximation and detail coefficients may be efficiently coded using a modified version of SPIHT. This chapter describes how these ideas have been implemented in an experimental video codec.

A wavelet-based video codec using **S**et **P**artitioning and the **S**elective **Pred**iction of **DC** frames (SPRED)[1] was implemented in software. Most of the code for SPRED was written in *MATLAB*, while some functions were coded in *C*.[2] Code for the implementation of Arithmetic Coding [15] was obtained from [16].

6.1 Implementation Details

Compression Specifications

The SPRED codec allows for the degree of compression desired by a user to be specified in terms of bit-rate (in bits per second). Currently, only a constant bit-rate (CBR) is permitted, though an extension to variable bit-rate (VBR) could easily be implemented. For a given target bit-rate, BR_{Video}, the number of bits allocated to each GOF, B_{GOF}, is given by:

$$B_{\mathrm{GOF}} = \frac{N_F}{N_{fps}} BR_{\mathrm{Video}}$$

[1] The acronym should read *SPSPRED*, but has been "compressed" to *SPRED*

[2] The source code is included on the CD-ROM.

where N_F is the number of frames in the GOF and N_{fps} represents the video frame rate.

Each GOF (with the exception of the first) is allocated B_{GOF} bits. The first GOF is instead allocated $B_{\text{GOF}} - 96$ bits. This is because each compressed file has a 12 byte (96 bit) *header*.[3] In order to ensure that the specified bit-rate is maintained, the first GOF is allocated 96 fewer bits than the other GOFs, to compensate for the additional 96 bits of the header.

Video Format

The SPRED *encoder* was designed to read from a (header-less) uncompressed video file. A video file was required to consist of a sequence of colour images, concatenated into one file. Each image was assumed to be in raster-scanned YC_bC_r format with each of the two chrominance components *half* the resolution of the luminance component. Each luminance or chrominance sample was specified with eight-bit precision in the range 0 to 255. The *decoder* was designed to produce a (decompressed) output file with the same specifications.

GOF size and Decomposition Levels

As highlighted in Section 4.2, the number of frames in a GOF influences the coding delay. It was therefore decided to use a four-frame GOF and two levels of temporal decomposition. For a frame rate of 30 fps, a four-frame GOF results in a delay of 133 ms, which is generally acceptable for interactive communication [2]. The number of levels of (luminance) *spatial* decomposition was left up to the user to specify. Since two chrominance components are each half the resolution of the luminance component, one *fewer* level of spatial decomposition was used for the chrominance components.

Filters

The choice of filters in a wavelet decomposition is important. The bi-orthogonal 9/7 filters of [1] are widely used in image compression applications. Consequently, it was decided to use this set of filters for the 2D (spatial) wavelet transform. The Haar filters [32] were chosen for the *temporal* dimension. This is because these (2-tap) filters allow for a relatively efficient implementation, and do not require any boundary extension.

[3]The header contains information such as frame rate, spatial resolution, bit-rate, etc.

The Prediction Map

If selective prediction is to be used, a binary prediction map, $\mathcal{M}$, (as defined in Section 4.3.2) must be coded together with each GOF. The encoder does this by *prefixing* the values in $\mathcal{M}$ to the SPIHT bit-stream (obtained by coding the GOF). Note that each value in $\mathcal{M}$ is specified by one bit. In addition, the order in which the values in $\mathcal{M}$ are specified is the same "zig-zag" pattern as that used for the spatial approximation coefficients, as described below.

Ordering of Spatial Approximation Coefficients

The spatial approximation coefficients in a frame are represented by the list $\mathcal{H}$ (as defined in Section 4.3.2). Generally, there is some correlation between neighbouring coefficients in $\mathcal{H}$. Consequently, it was decided to arrange the coefficients in $\mathcal{H}$ in a "zig-zag" order so that there would be a similar correlation between neighbouring elements in the list $\mathcal{H}$. The *adaptive* Arithmetic Coder would be able to exploit this correlation in order to achieve an improved compression performance.

The "zig-zag"ordering of coefficients is similar to that used for blocks of DCT coefficients in JPEG [21]. The ordering sequence for a 4×7 grid of approximation coefficients is given in Table 6.1 as an example.

Table 6.1: An example of "zig-zag" ordering for a 4×7 grid.

1	2	6	7	14	15	22
3	5	8	13	16	21	23
4	9	12	17	20	24	27
10	11	18	19	25	26	28

Arithmetic Coding

C source code for an adaptive arithmetic coder was obtained from [16]. The bit-based coding model was used to compress the bit-stream generated by SPIHT. An eight-bit context was specified for the adaptive model.

Prevention of Error Propagation

It was decided to use selective prediction for *five in every six* GOFs, with the *first* GOF in every six using *no* prediction. This prevents any channel errors which may occur from propagating for more than 24 frames.

6.2 The Structure of the *SPRED* Codec

6.2.1 The *SPRED* Encoder

Given an uncompressed video sequence, the encoder loads four frames into memory at a time as a GOF. For a colour sequence there are three GOFs - one for the luminance component and two for the chrominance components. The version of the codec implemented in software assumes that the C_r and C_b components are each half the *spatial* resolution of the Y component. As illustrated in Figure 6.1, each (4-frame) GOF is processed as follows:

- All pixels[4] are offset by a value of -128. This ensures that approximation coefficients are more likely to be clustered around zero.
- The GOF is transformed *temporally* using the 1D wavelet transform.
- Each frame in the group is then transformed *spatially* using the non-standard 2D wavelet transform. A *symmetric* boundary extension method is used. Note that the number of spatial decomposition levels is one less in the case of the two chrominance components.
- Following this, selective prediction is used to modify the DC frame. A corresponding binary prediction map is also generated. The binary prediction map is calculated using the *luminance* component only. The same map is then used for the two chrominance components. (Note that prediction is *not* used for one in every six DC frames, as explained in Section 6.1.)
- Modified 2D SPIHT for Video is then used to encode the 3D approximation and detail coefficients in the GOF, resulting in an interleaved bit-stream.
- The bit-stream is further compressed using *binary arithmetic coding* [15]. (The Prediction Map is also encoded at the front of the bit-stream.)
- The resulting bit-stream is truncated to exactly B_{GOF} bits so that the user-specified bit-rate is attained.[5]
- In a *partial feedback loop*, arithmetic decoding, inverse 2D SPIHT, and selective prediction using the prediction map are performed to decode the DC frame. This is necessary in order to ensure that the encoder and decoder both use the same reference DC frame during the selective prediction process.

[4]The encoder assumes that Y, C_r and C_b samples lie in the range $[0, 255]$.

[5]Note that this desired bit-rate is included in the header of the compressed file, so that it is also available to the decoder.

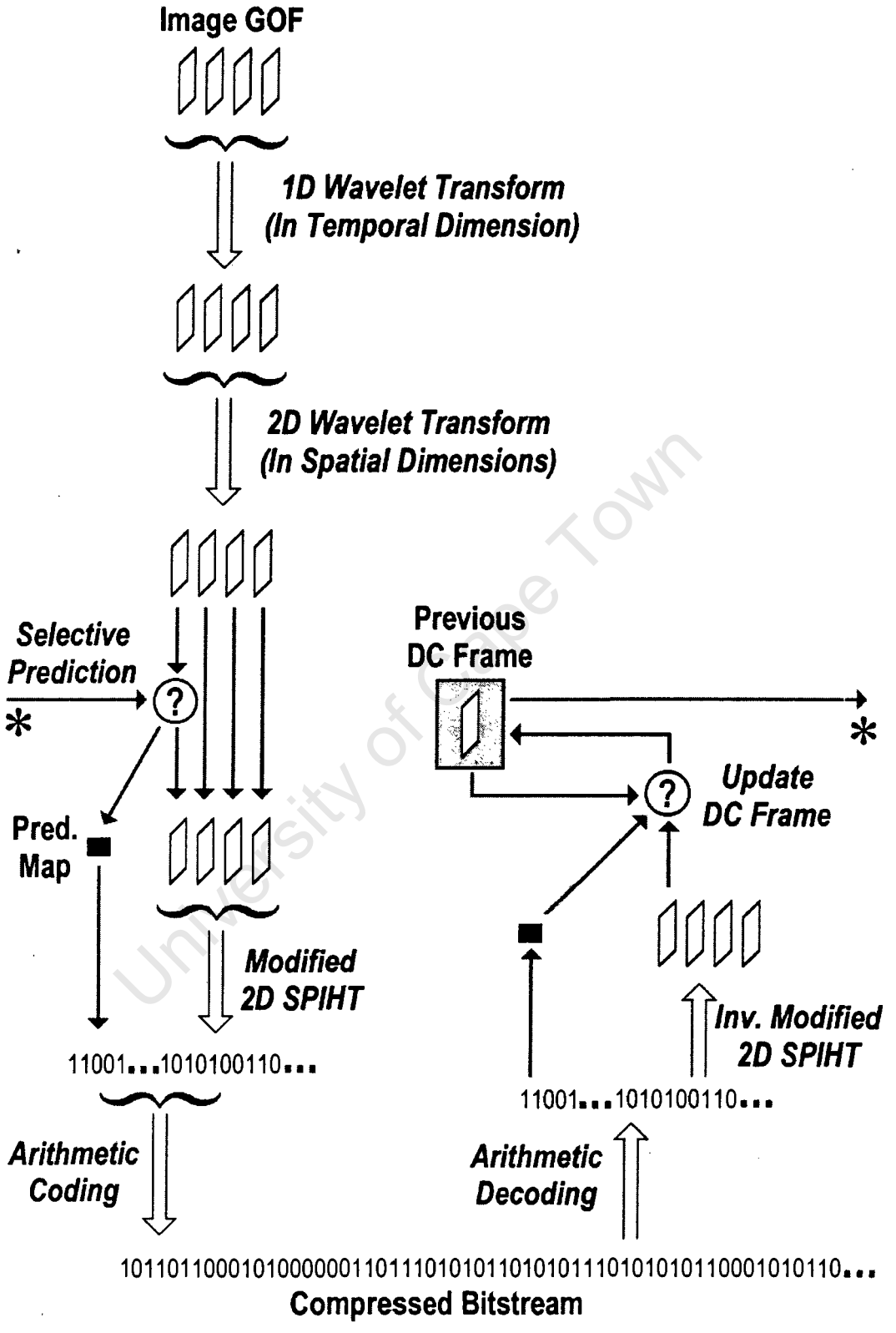


Figure 6.1: The structure of the SPRED Encoder. (The feedback loop is shaded grey.)

6.2.2 The *SPRED* Decoder

The SPRED decoder performs the same operations as the encoder, but in reverse order (and *without* the partial feedback loop). As illustrated in Figure 6.2, the decoding process proceeds as follows:

- The decoder reads in B_{GOF} bits[6], corresponding to the compressed information describing each GOF.
- This bit-stream is then decoded using an arithmetic decoder. The *Prediction Map* and the SPIHT bit-stream can thus be extracted.
- Following this, the SPIHT bit-stream is decoded using the inverse of the modified 2D SPIHT coding process. A group of frames representing 3D approximation and detail coefficients results. The first frame in this group is a selectively-predicted DC frame. (In the case of a colour sequence, three GOFs are produced.)
- Using the Prediction Map and the previous DC frame, the decoder is able to restore the DC frame. This restored frame is also stored so as to be used in the decoding of the *next* GOF.
- The inverse 2D wavelet transform is then applied to each frame in the GOF. For the two chrominance components, one *less* level of *spatial* decomposition is required than in the case of the luminance component.
- Following this, the inverse 1D wavelet transform is applied to the GOF in the *temporal* dimension.
- Finally, the pixel values are offset by 128, and are examined to ensure that the Y, C_r and C_b components of the decoded frames lie in the range $[0, 255]$. If not, the samples are thresholded as required.

[6]Or, in the case of the initial portion of the bit-stream, the 96 bit header and $B_{\text{GOF}} - 96$ bits.

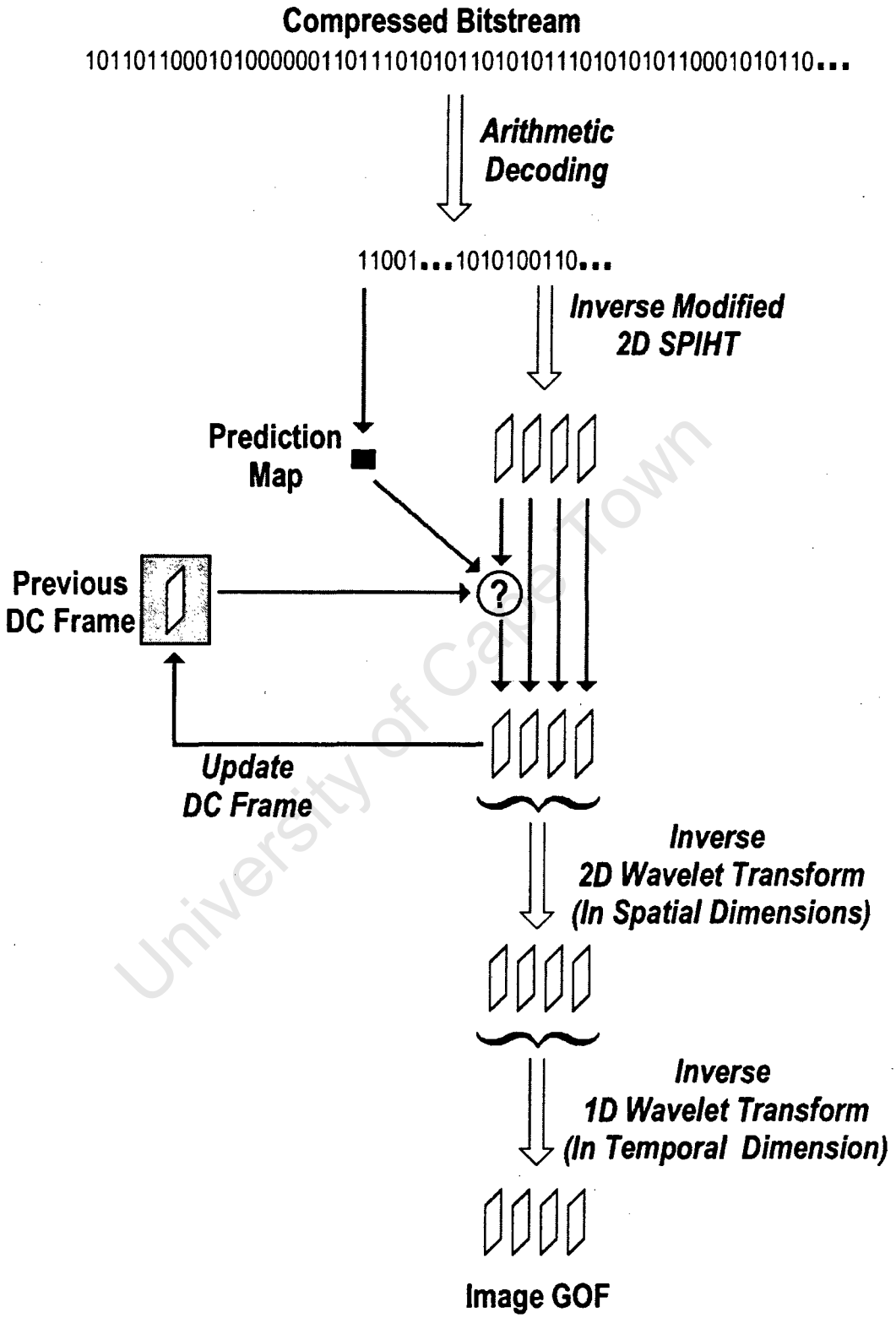


Figure 6.2: The structure of the SPRED Decoder.

Chapter 7

A Performance Analysis of the *SPRED* Video Codec

This section describes the performance of SPRED relative to implementations of MPEG-1 and MPEG-2. It also demonstrates two of the main advantages of SPRED - its progressive transmission capabilities and its performance without entropy coding.

7.1 The Test Sequences

Two CIF[1] and five SIF[2] sequences were used in testing - each in uncompressed YC_rC_b format, with 4:2:0 chrominance sub-sampling. The sequences used contain different degrees of object motion and spatial detail. In addition, they were filmed using varying types of camera motion. A short description of each sequence is provided below, together with a list of the various scenes in Table 7.1. Note that a scene *transition* was assumed to be either a change in subject or in the type of camera motion (e.g. from still to panning).

Claire This is a typical "head and shoulder" sequence. It consists of 164 frames of a woman talking, in front of a uniform, blue background. There is no background motion, but some motion of the woman's head, mouth and eyes. The camera is fixed throughout. This CIF-format sequence was obtained from [20].

Salesman This is another "head and shoulder" sequence. It depicts a man sitting at a desk and talking. There is also some motion of the man's arm and an object which he is holding.

[1]Common Intermediate Format: 352 × 288 pixels per frame, 30 frames per second (fps).
[2]Standard Interchange Format: 352 × 240 pixels per frame, 30 fps.

The presence of a bookshelf and some plants contribute to a significant amount of background detail. The camera is fixed throughout. This CIF-format sequence was obtained from [20].

Bike This 148-frame sequence depicts a man on a motor-bike ramping off the top of a high wall and landing on the ground below. When the bike lands, several high-intensity sparks are produced. Throughout the sequence, the general direction of motion of the bike is towards the camera. The camera pans slowly throughout the sequence, in order to track the bike. This SIF-format sequence was obtained from [20].

Football 1 This sequence depicts part of a game of American Football. For the first 70 frames the camera is fixed. Several players (moving very fast) come into view from various angles. In frames 71 to 124, the camera begins to pan slowly, and more players enter the field of view (though they move more slowly). This SIF-format sequence was obtained from [4].

Football 2 This sequence depicts a *different* part of the same game of American Football described above. For the first 84 frames, the camera is fixed. During this time there is some object motion, though it is relatively slow. Frames 85 to 208 are characterised by fast object motion and camera panning. During part of the camera pan (approximately frames 140 to 170), the scene is dominated by the grass in the background, so that there is relatively little spatial detail during this interval. This SIF-format sequence was obtained from [20].

Flower Garden This 112-frame sequence depicts a house in the background, with a tree and a garden of flowers in the foreground. The multi-coloured flowers in the foreground convey a substantial amount of spatial detail. For the entire sequence, the camera exhibits translational motion, as though the scene were filmed from a moving vehicle. This SIF-format sequence was obtained from [4].

Table Tennis This sequence depicts a game of table tennis between two players. For the first 24 frames the camera is still - focussed on the arm and bat of Player One, as he bounces the ball. The camera then zooms out (from frames 25 to 67) until Player One is fully in view. There is then a cut, and Player Two can be seen during frames 68 to 97. During this time the camera is still, and there is relatively little background spatial detail. This is followed by another scene cut. During frames 98 to 112, the camera (which is still) once again focuses on Player Two. However, part of the background of this scene consists of a cartoon, in the form of a poster on the wall. The poster contributes a significant amount of spatial detail to the scene. This SIF-format sequence was obtained from [4].

Table 7.1: A List of the Scenes in the Test Sequences used

Sequence	Frames	Type of Camera Motion	Translational Object Motion	Degree of Spatial Detail
Claire	1-164	Still	No	Low
Salesman	1-100	Still	Yes	High
Bike	1-148	Pan	No	Medium
Flower Garden	1-112	Translation	No	High
Football 1	1-70	Still	No	Medium
Football 1	71-124	Pan	Yes	High
Football 2	1-84	Still	Yes	Medium
Football 2	85-208	Pan	No	Medium
Table Tennis	1-24	Still	Yes	Low
Table Tennis	25-67	Zoom Out	Yes	Medium
Table Tennis	68-97	Still	No	Low
Table Tennis	98-112	Still	No	Medium

7.2 The "Compression vs. Quality" Performance of SPRED

This section describes the tests performed in order to evaluate SPRED, and gives an indication of the codec's performance. It shows the objective performance of SPRED relative to MPEG (using the PSNR measure). Some sample frames are also provided as an example of the subjective quality of the sequences compressed with SPRED or MPEG. Following this, the performance of SPRED is discussed on a scene-by-scene basis.

7.2.1 Test Procedure

The SPRED codec was tested on the seven standard test video sequences, and compared with the performance of MPEG-1 and MPEG-2 on these sequences. A software MPEG codec (MPEG-2 version 1.2, July 1996) was obtained from [17]. With this program, it was possible to encode a video sequence using either the MPEG-1 or MPEG-2 standard, at a specified bit-rate.[3]

Each sequence was compressed at the desired bit-rate using MPEG-1, MPEG-2 and SPRED. The tests were performed at constant bit-rates of either 1.0 or 1.5 Mbps.[4] The resultant bit-streams were then decoded to produce a video sequence. The quality of each decompressed sequence was evaluated using the mean PSNR measure, as described in Section 2.4.1. Some frames from the decompressed sequences are also shown, in order to demonstrate the

[3]Samples of the various options used for the MPEG codec are provided in Appendix C.
[4]1,048,576 or 1,572,864 bits per second.

subjective quality obtained with each of the three codecs.

7.2.2 Objective Performance

MPEG-1 vs. MPEG-2

Table 7.2 shows the performance of SPRED and MPEG-2 *relative* to MPEG-1 for the seven sequences. From this table, it is interesting to note that MPEG-1 out-performed MPEG-2 slightly in all but one of the test cases. This is probably because MPEG-1 was optimised for SIF-resolution video at bit-rates of 1.1 Mbps, whereas MPEG-2 was designed to cater primarily for higher resolution sequences at bit-rates of around 4 Mbps. Thus, for most of the subsequent discussion, the performance of SPRED relative to MPEG-1 (rather than MPEG-2) is considered.

The CIF sequences

The SPRED codec showed a substantial improvement in quality over MPEG-1 for the two CIF sequences. This can be ascribed to the fact that "Claire" and "Salesman" exhibit relatively little object motion and no camera motion. This lack of motion is well exploited by the selective prediction algorithm.

Recall that for one in every *six* GOFs, no prediction was used.[5] Consequently, for the low-motion "Claire" and "Salesman" sequences, a drop in quality was observed in those GOFs which for which selective prediction was not used. This drop in quality is evident every 24 frames in Figures B.1 and B.2.

The SIF sequences

For the five SIF sequences, the *luminance* performance of SPRED and MPEG-1 varied significantly. For example, MPEG-1 performed substantially better than SPRED for the "Flower Garden" sequence, while the opposite was true in the case the "Table Tennis" sequence. For most of the other SIF sequences, SPRED exhibited a slightly superior luminance performance.

MPEG-1 showed consistently better results for both the *chrominance* components of the SIF sequences. However, it is worth noting that the chrominance components are each half the resolution of the the luminance component, and *together* the two chrominance components constitute only *one third* of the information content of the sequence.

[5]This was done in order to limit error propagation in a noisy channel.

Table 7.2: A Comparison of the Performance of the MPEG-1, MPEG-2 and SPRED Codecs at constant bit-rates. *Mean PSNR per Frame* values for MPEG-1 are given, while for MPEG-2 and SPRED, improvements in *Mean PSNR per Frame* over MPEG-1 are indicated.

Video Sequence	Bit-rate (Mbps)	Component	MPEG-1 (dB)	MPEG-2 (± dB)	SPRED (± dB)
CLAIRE		**Y**	**44.82**	**-0.00**	**+2.65**
164 frames	1.0	Cb	45.20	-0.15	+1.56
CIF format		Cr	47.26	-0.26	+1.02
SALESMAN		**Y**	**36.78**	**-0.08**	**+2.87**
100 frames	1.0	Cb	41.95	-0.10	+1.29
CIF format		Cr	44.33	-0.04	+1.01
BIKE		**Y**	**38.48**	**-0.02**	**+0.09**
148 frames	1.0	Cb	41.70	-0.07	-0.24
SIF format		Cr	42.41	-0.07	-0.51
FOOTBALL 1		**Y**	**27.27**	**-0.20**	**-0.20**
124 frames	1.0	Cb	32.22	-0.12	-1.35
SIF format		Cr	34.39	-0.07	-0.94
FOOTBALL 2		**Y**	**30.62**	**-0.21**	**+0.90**
208 frames	1.0	Cb	34.17	-0.14	-0.18
SIF format		Cr	38.07	-0.09	-0.21
BIKE		**Y**	**40.05**	**+0.01**	**+0.98**
148 frames	1.5	Cb	42.50	-0.10	-0.16
SIF format		Cr	43.27	-0.12	-0.29
FOOTBALL 1		**Y**	**29.10**	**-0.14**	**+0.09**
124 frames	1.5	Cb	33.46	-0.10	-0.79
SIF format		Cr	35.26	-0.07	-0.67
FOOTBALL 2		**Y**	**32.48**	**-0.16**	**+0.93**
208 frames	1.5	Cb	35.51	-0.13	-0.56
SIF format		Cr	38.93	-0.09	-0.41
FLOWER GARDEN		**Y**	**27.28**	**-0.17**	**-3.98**
112 frames	1.5	Cb	30.20	-0.14	-3.19
SIF format		Cr	32.24	-0.10	-2.15
TABLE TENNIS		**Y**	**33.85**	**-0.16**	**+1.65**
112 frames	1.5	Cb	39.60	-0.13	-0.15
SIF format		Cr	38.92	-0.05	-0.95

Sequences tested at different bit-rates

Three of the sequences ("Bike", "Football 1" and "Football 2") were tested at both 1.0 and 1.5 Mbps. At the lower bit-rate, SPRED exhibited a superior luminance performance to MPEG-1 in two of the three cases; while at the higher bit-rate, SPRED out-performed MPEG-1 in terms of luminance for all three sequences. MPEG-1 demonstrated a superior chrominance quality at both bit-rates, although its advantage over SPRED was generally less substantial at the higher bit-rate. This can probably be ascribed to the fact that the lower of the two bit-rates is closer to the MPEG-1 optimal video bit-rate of 1.1 Mbps.

The performance of SPRED is perhaps more remarkable when one considers that it does not use any of the sophisticated *motion estimation* techniques employed by MPEG, and that the five SIF sequences consist of scenes with significant amounts of object and camera motion.

A more detailed scene-by-scene analysis of the relative performance of SPRED and MPEG follows in Section 7.2.4.

7.2.3 Subjective Performance

As highlighted in Section 2.4.1, objective image quality (such as that measured using PSNR) does not necessarily correlate with the quality perceived by a human observer. Due to time constraints, it was not possible to conduct subjective tests with a reasonably large sample in order to measure the relative performance of SPRED and MPEG. However, this section provides some sample frames from two of the test sequences[6] in order to illustrate some of the characteristics of each method (i.e. MPEG-1, MPEG-2 and SPRED).

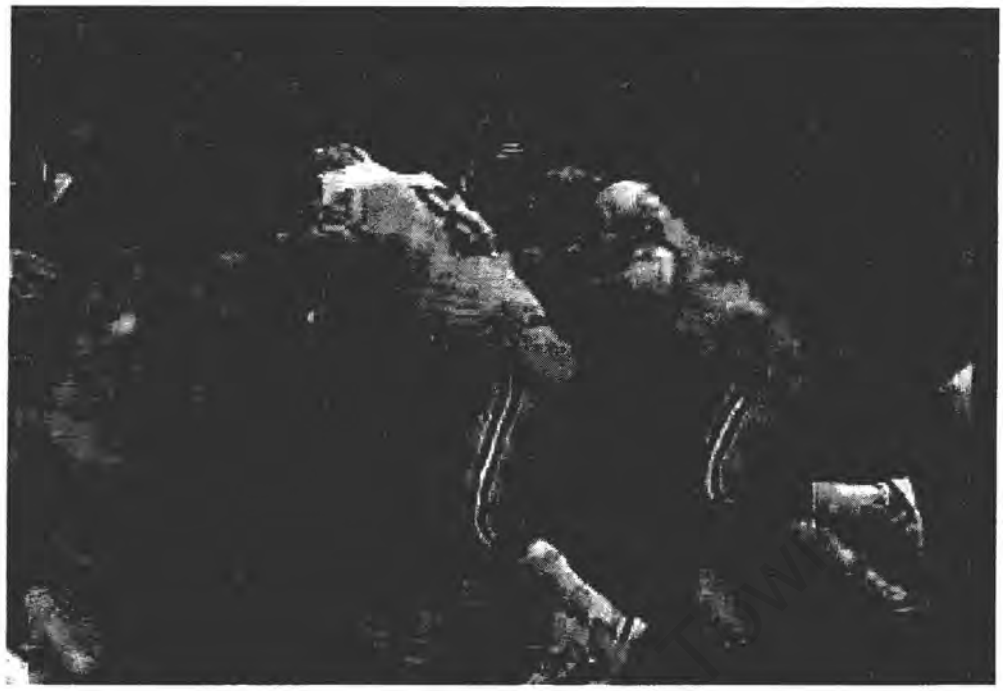
The "Football 2" sequence

Figure 7.1 depicts frame 119 of the "Football 2" sequence. The frames from the MPEG-1 and MPEG-2 sequences exhibit some significant blocking artifacts, which are characteristic of MPEG. The frame from the sequence compressed with SPRED shows some blurring, particularly in the bottom left of the image. However, in this frame, SPRED generally appears to provide better preservation of detail than MPEG. For example, the numbers on the players' jerseys as well as their socks and boots are noticeably more distinct in the frame from the SPRED sequence. The PSNR rankings for these frames are: SPRED (30.6 dB), MPEG-1(28.1 dB) and MPEG-2(27.9 dB).

[6]Colour videos of most of the sequences (original and compressed) are provided on the CD-ROM.

(a) Original

(b) MPEG-1 (PSNR: 28.16 dB)



(c) MPEG-2 (PSNR: 27.91 dB)

(d) SPRED (PSNR: 30.56 dB)

Figure 7.1: Luminance component of Frame 119 of "Football 2" compressed at 1.0 Mbps and 30 fps.

The "Flower Garden" sequence

Figure 7.2 shows frame 40 from the "Flower Garden" sequence. The frames from the two MPEG sequences represent a reasonably accurate reconstruction of the original. The most noticeable artifact is the presence of ringing around some of the trees in the background. The frame from the sequence compressed using SPRED shows a substantial amount of blurring. This is particularly noticeable with the trees, in both the foreground and background. The PSNR rankings for these frames are: MPEG-1(28.0 dB), MPEG-2(27.8 dB) and SPRED(23.6 dB).

It is encouraging to note that for the two examples given above (with frames from the "Football 2" and "Flower Garden" sequences) the PSNR values seem to provide a reasonable guide to image quality.

7.2.4 A Scene-based Performance Analysis

The objective results and the sample images presented above show that the performance of SPRED and MPEG can vary significantly, depending on the sequence being compressed. This section attempts to illustrate the types of scenes for which SPRED is most appropriate as a compression method.

As demonstrated above, SPRED performed well on the "Salesman" sequence, but rather poorly in the case of the "Flower Garden" sequence. Both of these sequences contain significant spatial detail, and little or no *object* motion. The only substantial difference between the two is the *camera* motion (i.e. *still* for "Salesman" and *translational* for "Flower Garden"). It was therefore decided to analyse scenes according to the type of camera motion with which they were filmed. Refer back to Table 7.1 for a list of the scenes.

The graphs in Figures 7.3, 7.4, 7.5 and 7.6 show the average Mean Square Error (MSE) per frame for each of the sequences. Note that the MSE (unlike the PSNR measure) should ideally be minimised. The MSE value depicted in each of the bar graphs is a *weighted* sum of the MSE for the luminance and chrominance components. The luminance (Y) MSE is weighted by a factor of $\frac{2}{3}$, while each of the chrominance components (C_b and C_r) contribute a MSE weighted by $\frac{1}{6}$.[7]

[7]The weighting factors are proportional to the ratios of luminance and chrominance samples.

(a) Original

(b) MPEG-1 (PSNR: 28.03 dB)

(c) MPEG-2 (PSNR: 27.75 dB)

(d) SPRED (PSNR: 23.56 dB)

Figure 7.2: Luminance component of Frame 40 of "Flower Garden" compressed at 1.5 Mbps and 30 fps.

Scenes filmed with a *Still Camera*

Figure 7.3 depicts the average MSE per frame for each of the seven scenes[8] filmed with a still camera. Significantly, SPRED showed the lowest MSE for *all* of the nine cases. The improvement provided by SPRED over MPEG-1 was most evident for the "Claire", "Salesman" and the three "Table Tennis" scenes.

These results demonstrate that SPRED is well-suited to coding scenes filmed with a still camera. Generally, the performance of SPRED is closely related to the proportion of the scene occupied by moving objects. This is because *selective prediction* allows for those regions of a scene not occupied by moving objects (e.g. the background) to be accurately predicted form one GOF to the next.

Significantly, SPRED showed superior performance to MPEG-1 for frames 1 to 70 of "Football 1". This is despite the fact that this scene contains a substantial amount of object motion, and that SPRED uses none of the sophisticated *motion estimation* techniques employed by MPEG.

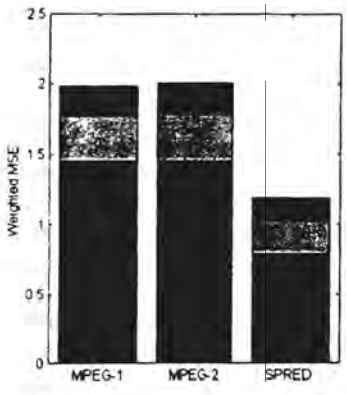
The dominant form of object motion in the "Salesman" sequence and in the first 24 frames of "Table Tennis" is *translational* motion. Remarkably, SPRED demonstrated substantially lower MSE for these sequences than MPEG, even though the motion vectors use by MPEG were specifically designed to describe translational motion. From these results, it appears as though the selective prediction method employed by SPRED out-performs the motion estimation technique of MPEG, for relatively low-motion sequences filmed with a still camera.

Scenes filmed with a *Panning Camera*

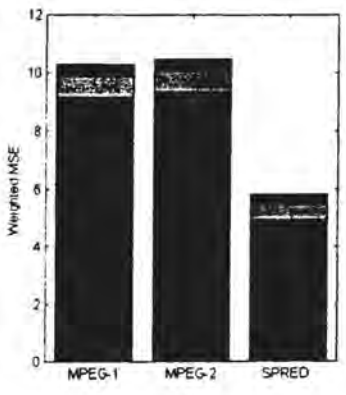
Figure 7.4 shows the average MSE per frame for each of the three scenes[9] filmed with a panning camera. MPEG-1 showed superior performance in three of the cases, while SPRED out-performed MPEG for the other three cases.

For the "Bike" sequence, all three coding methods (i.e. MPEG-1, MPEG-2 and SPRED) showed similar MSE values when tested at a bit-rate of 1.0 Mbps, though MPEG-1 was marginally superior. However, when tested at a bit-rate of 1.5 Mbps, the MSE of the SPRED sequence showed a significant improvement over those of MPEG-1 and MPEG-2 at the same bit-rate. The motion of the motor-bike in the scene is generally toward the camera. However this type of motion cannot be well-described by MPEG, which uses motion vectors to account for translational motion.

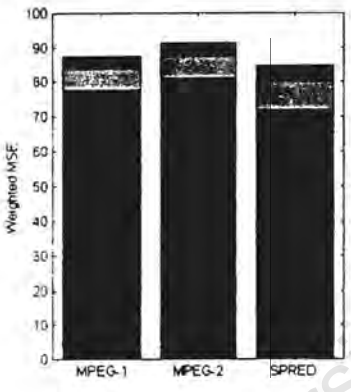
[8]There are *nine* graphs because the two "Football" scenes were each tested at two bit-rates.
[9]There are *six* graphs because each of the scenes was tested at two bit-rates.



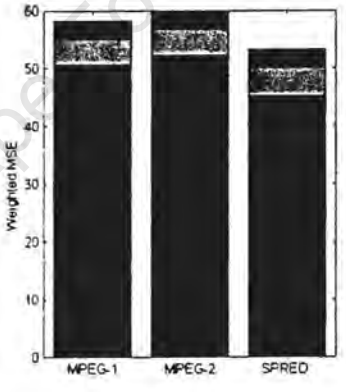
(a) Claire, 1.0 Mbps



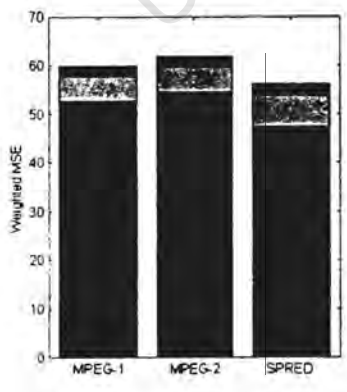
(b) Salesman, 1.0 Mbps



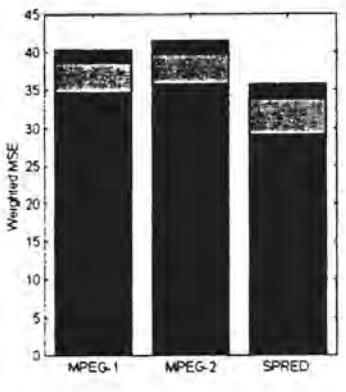
(c) Football 1 (frames 1-70), 1.0 Mbps



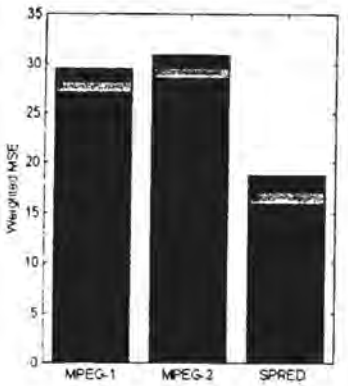
(d) Football 1 (frames 1-70), 1.5 Mbps



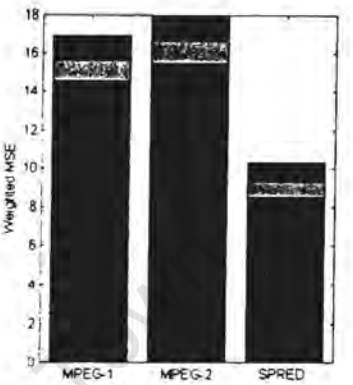
(e) Football 2 (frames 1-84), 1.0 Mbps



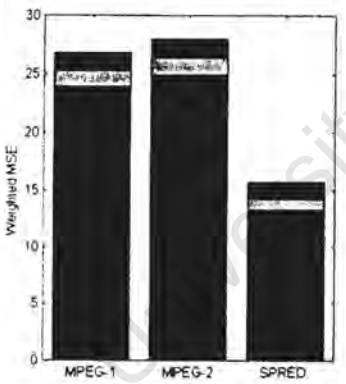
(f) Football 2 (frames 1-84), 1.5 Mbps



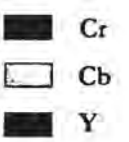
(g) Table Tennis (frames 1-24), 1.5 Mbps



(h) Table Tennis (frames 68-97), 1.5 Mbps

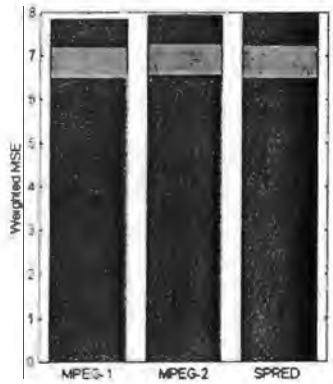


(i) Table Tennis (frames 98-112), 1.5 Mbps

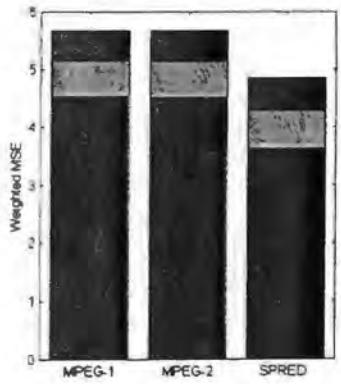


(j)

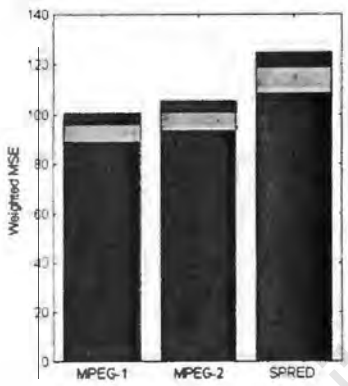
Figure 7.3: The weighted Mean Square Error for the *still camera* scenes



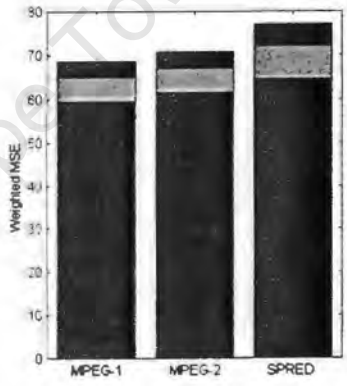
(a) Bike, 1.0 Mbps



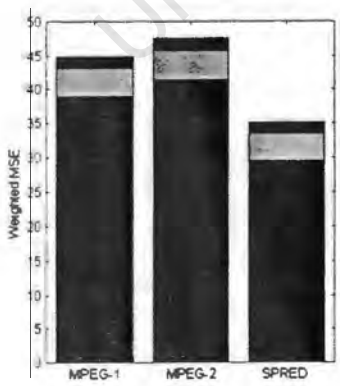
(b) Bike, 1.5 Mbps



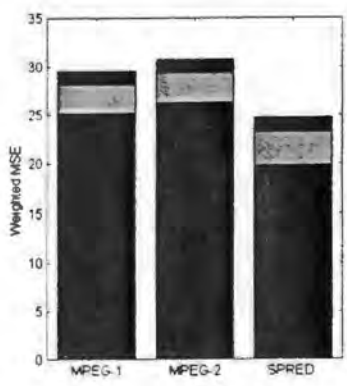
(c) Football 1 (frames 71-124), 1.0 Mbps



(d) Football 1 (frames 71-124), 1.5 Mbps



(e) Football 2 (frames 85-208), 1.0 Mbps



(f) Football 2 (frames 85-208), 1.5 Mbps

Figure 7.4: The weighted Mean Square Error for the *panning camera* scenes

Frames 71 to 124 of the "Football 1" sequence contain a substantial degree of object motion, with only a relatively small portion of the background visible. In addition, much of the object motion is translational, and can thus be well-represented with motion vectors. This accounts for the superior performance of MPEG for this scene (as measured in terms of MSE).

The scene comprising frames 98 to 208 of the "Football 2" sequence includes a significant amount of fast camera panning, as well as substantial object motion. Much of the object motion is not translational. Consequently, SPRED provides significantly better performance (in terms of MSE) than MPEG at both bit-rates.

The above results demonstrate that neither SPRED nor MPEG perform *consistently* better or worse on sequences filmed with a panning camera. A significant factor influencing the relative performance of SPRED and MPEG is the degree of translational motion exhibited by either the background or the objects in a scene. It was observed that MPEG demonstrated a generally superior compression performance for those scenes containing a significant degree of *translational* motion.

Scenes filmed with a *Zooming Camera*

Only *one* of the seven test sequences was filmed with a camera *zooming out*. Frames 25 to 67 of the "Table Tennis" sequence depict a man bouncing a ball up and down with his bat (i.e. translational motion) while the camera zooms out to reveal a scene with a fair amount of spatial detail. Figure 7.5 indicates that MPEG-1 performed significantly better than SPRED for this scene.

The inferior performance of SPRED can be explained by noting that the camera zooming prevented accurate prediction of the DC frame, since there was no static background. The translational object motion was also relatively well-represented by MPEG motion vectors.

Scenes filmed with a *Translating Camera*

The "Flower Garden" sequence consists of a scene with high spatial detail, viewed by a translating camera, moving at a fairly constant rate. Because the flowers in the foreground are close to the camera they *appear* to move at a relatively fast rate. In contrast, the houses in the background appear to move at a slower rate. This type of scene can be well-modelled by the block-based motion estimation techniques of MPEG, since motion vectors are calculated for each region (block) of a scene.

In contrast, the selective prediction algorithm used by SPRED is *not* well suited to the coding of such a scene. Because the whole scene is constantly moving, there can be little

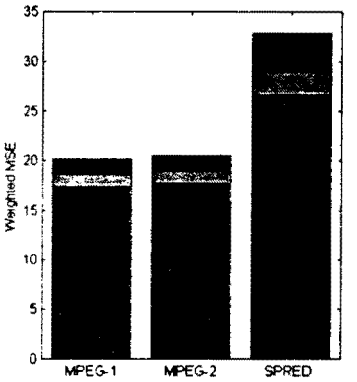


Figure 7.5: The weighted Mean Square Error for the *zooming camera* scene: Table Tennis (frames 25-67), 1.5 Mbps

or no accurate prediction of the DC frame of a transformed GOF. In addition, the *temporal* correlation between consecutive frames in such a constant-motion sequence is significantly reduced if the scene contains high spatial detail.

As shown in Figure 7.6, MPEG-1 performs substantially better than SPRED for this scene, achieving less that half the MSE obtained with SPRED.

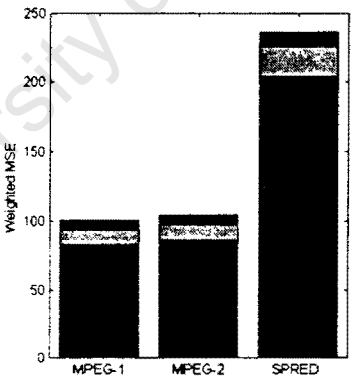


Figure 7.6: The weighted Mean Square Error for the *translating camera* scene: "Flower Garden", 1.5 Mbps

Summary

By considering the performance of SPRED for the scenes tested, it is possible to establish which types of scenes SPRED is most suited to coding. Based on the results listed above (for

scenes filmed with a still, panning, zooming or translating camera) the following trends are evident:

- SPRED is well-suited to compressing scenes filmed with a stationary camera. This was shown to be true irrespective of the degree of spatial detail in the background. The amount and type of object motion does influence the performance of SPRED. Nevertheless, provided that the background constitutes a significant portion of the scene, the selective prediction algorithm enables SPRED to efficiently compress scenes containing substantial object motion.
- SPRED performs less well in coding scenes filmed with a moving camera. A significant factor influencing the ability of SPRED to efficiently compress such scenes is the degree of spatial detail present in the background. If the background of a scene contains substantial spatial detail, this will reduce the *temporal* correlation between successive frames, and thus limit the effectiveness of the wavelet transform in the temporal domain. However, if the scene background contains low or medium spatial detail, the correlation between successive frames will still be substantial. Under such conditions, both the temporal wavelet transform and the selective prediction algorithm will enable SPRED to achieve a significant degree of compression.
- The type of motion in a scene can significantly influence the *relative* performance of SPRED and MPEG. This is because the motion vectors used by MPEG have been designed to model *translational* motion. Thus, MPEG would generally show superior performance over SPRED in scenes with substantial translational motion.[10]

7.3 Some Additional Properties of SPRED

This section examines some of the properties and features of the SPRED codec, in particular: the performance of SPRED without entropy coding; the effect of the number of levels of spatial decomposition; and the progressive transmission capabilities of SPRED.

7.3.1 Performance without Entropy Coding

Entropy coding plays a crucial role in most image and video codecs (including MPEG). However, as a result of its use of SPIHT, SPRED generally achieves a significant degree of compression even *without* entropy coding. The advantage of this is that if an encoder is unable

[10]The motion estimation techniques used by MPEG are computationally expensive, and this is sometimes viewed as a significant drawback of the codec.

to perform arithmetic coding in real-time, a sequence can still be compressed significantly. An example is illustrated in Figure 7.7 for the "Salesman" sequence.

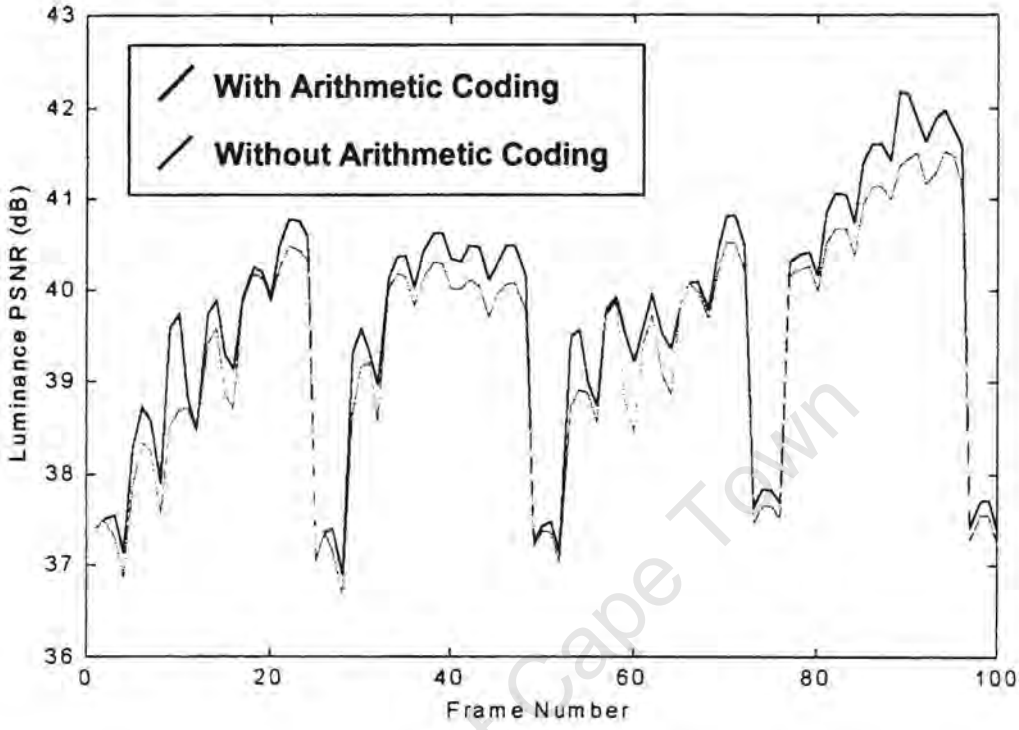


Figure 7.7: The luminance component of the "Salesman" sequence at 1.0 Mbps and 30 fps, with and without Binary Arithmetic Coding.

7.3.2 Levels of Spatial Decomposition

The SPRED codec allows the user to specify the number of levels of spatial decomposition applied to each luminance frame.[11] Table 7.3 shows the effect of varying the number of levels of spatial decomposition for two of the test sequences.

From Table 7.3 it is evident that both the "Salesman" and "Bike" sequences perform slightly better when more levels of spatial decomposition are used. This trend is also generally true of *images* compressed using EZW [28] or SPIHT [25].

[11] Recall that one fewer level of decomposition is used for the chrominance components.

Table 7.3: The performance of SPRED for different levels of spatial decomposition.

Video Sequence	Decomposition Levels (Y)	Y PSNR (dB)	Cb PSNR (dB)	Cr PSNR (dB)
Salesman	5	39.65	43.24	45.34
Salesman	4	39.62	43.17	45.15
Bike	4	38.57	41.46	41.90
Bike	3	38.23	41.23	41.64

7.3.3 Progressive Transmission

The progressive transmission capabilities of SPRED were tested using three SIF sequences. The sequences were all *decoded* at bit-rates of 1.0 Mbps from bit-streams which had been *encoded* at 1.5 Mbps. This was done by decoding the first $\frac{2}{3}$ of the bit-stream for each GOF and discarding the remaining $\frac{1}{3}$. Table 7.4 shows the quality of the sequences decoded at the lower rate. As a comparison, the table also shows the quality of of each of the sequences when encoded at 1.0 Mbps and decoded at 1.0 Mbps.

Table 7.4: Progressive Transmission using SPRED

Video Sequence	Component	SPRED[a] (dB)	SPRED[b] (dB)
BIKE	**Y**	**38.57**	**38.46**
148 frames	Cb	41.46	41.34
SIF format	Cr	41.90	41.71
FOOTBALL 1	**Y**	**27.08**	**26.99**
124 frames	Cb	30.87	30.76
SIF format	Cr	33.45	33.42
FOOTBALL 2	**Y**	**31.52**	**31.28**
208 frames	Cb	33.99	33.55
SIF format	Cr	37.86	37.62

[a]Decoded at 1.0 Mbps from 1.0 Mbps Bit-stream
[b]Decoded at 1.0 Mbps from 1.5 Mbps Bit-stream

As shown in Table 7.4, there is a *small* loss in quality (of up to 0.3 dB for the luminance components) for the sequences decoded from the 1.5 Mbps bit-stream (relative to those decoded from the 1.0 Mbps bit-stream.) This can be ascribed to the fact that a slightly different reference frame is used during the *selective prediction* stage of the two encoding processes.

The *embedded* nature of SPRED bit-streams provides a significant practical advantage over

MPEG. As outlined in Section 5.1.3, a coding scheme which uses progressive transmission is particularly advantageous for the transmission of video over networks.

Chapter 8

Conclusion

Wavelet-based coding methods have been shown to be particularly effective for image compression. This thesis has described the design, implementation and analysis of SPRED, a video codec based on the 3D wavelet transform. This chapter summarises the advantages and disadvantages of SPRED, and suggests ways of improving the performance of the codec.

8.1 Conclusions

The structure of the SPRED codec is relatively simple. It consists of the following core methods applied to each group of frames:

- The 3D wavelet transform;
- Selective prediction of the DC frame;
- Modified 2D SPIHT;
- Entropy Coding (optional).

Compression Performance of SPRED

For the seven sequences used in testing, it was evident that the *type* of camera motion with which a sequence was filmed played a significant role in the relative degrees of compression achieved by SPRED and MPEG.

SPRED was shown to be particularly efficient for the coding of video sequences filmed with a *still camera*. This trend was observed across a range of scenes, consisting of various types

of spatial detail and *object* motion (including translational motion). SPRED was observed to perform substantially better than MPEG for the two "talking head" sequences used in testing.

The relative performance of SPRED and MPEG varied in the case of scenes filmed with a *moving camera*. For such scenes, the *type* of camera motion and the degree of spatial detail appeared to significantly influence the performance of SPRED. Scenes dominated by *translational motion* performed substantially better when compressed with MPEG. This is because MPEG motion vectors were designed to model translational motion. For scenes with other types of camera motion, SPRED performed well when the degree of spatial detail was low. However, MPEG demonstrated superior performance for scenes filmed with a moving camera *and* containing high spatial detail.

Features of SPRED

Significantly, SPRED uses no computationally-intensive motion estimation techniques like those employed by MPEG. It is also less reliant on entropy coding than MPEG. Adaptive entropy coding is sometimes considered too time-consuming for real-time encoding applications. However, SPRED can be used *without* arithmetic coding, with only a relatively small loss in video quality.

In addition, the SPRED encoder produces an *embedded* bit-stream, which allows for the progressive transmission of (compressed) video. Progressive transmission is particularly advantageous when video is transmitted over a congested network, since less important parts of a bit-stream may be discarded, resulting in only a partial loss in video quality.

Prediction of coefficients from one GOF to the next allows any channel errors which may arise, to propagate. However, the SPRED codec prevents the indefinite propagation of errors by *not* using selective prediction for one in every six GOFs.[1]

8.2 Recommendations

Additional research should be conducted in order to improve the performance of SPRED, both in terms of compression and efficiency. Some suggested improvements and areas of research are listed below.

- The current method of preventing error propagation could easily be modified to prevent

[1] For sequences with very little motion, this lack of prediction can manifest itself as a drop in quality every 24 frames.

a regular "dip" in video quality every 24 frames. This could be done by using *no prediction* for one in every six *spatial orientation trees* in each DC frame.[2]

- The performance of SPRED with different *temporal* filters could be tested. While filters other than the Haar filters may lead to improved compression performance, this would probably be at the expense of greater computation. Furthermore, in the case of a GOF with relatively few frames (e.g. four frames), a long filter might produce undesirable artifacts, due to boundary-extension edge effects.
- SPRED could be extended to handle interlaced frames effectively.
- An efficient implementation of the SPRED codec could be developed and coded in software so that it is able to compress at least low-resolution sequences in real-time.
- Additional tests could be performed to examine the performance of SPRED at higher and lower bit-rates. Tests could be performed on high-resolution sequences (for comparison with MPEG-2) and video-conferencing sequences (for comparison with H.263).
- SPRED has demonstrated impressive compression performance for scenes filmed with a still camera. This suggests that it may be particularly useful as a video-conferencing codec. Several modifications to SPRED could possibly be made to optimise it for video-conferencing applications.

[2]Rather than using no prediction for *all* trees in every sixth DC frame.

Bibliography

[1] M. Antonini, Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Trans. Image Processing*, pages 205–220, April 1992.

[2] Rita L. Atkinson, Richard C. Atkinson, Edward E. Smith, and Daryl J. Bem. *Introduction to Psychology.* Harcourt Brace Jovanovich, 10 edition, 1990.

[3] C.M. Brislawn. Classification of nonexpansive symmetric extension transforms for multirate filter banks. *Applied and Computational Harmonic Analysis*, 3:337–357, 1996.

[4] Rensselaer Polytechnic Institute Center for Image Processing Research. Image database. ftp://ipl.rpi.edu/pub/image2/sequence/sif/.

[5] Y. Chen and W.A. Pearlman. Three-dimensional subband coding of video using the zero-tree method. In *SPIE Symp. on Visual Communications and Image Processing*, 1996.

[6] Leonardo Chiariglione. Moving picture experts group (MPEG) home page. http://drogo.cselt.stet.it/mpeg.

[7] Michael L. Hilton, Bjorn D. Jawerth, and Ayan Senhupta. Compressing still and moving images with wavelets. *Multimedia Systems*, 2:218–227, 1994.

[8] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the IRE*, pages 1098–1101, September 1952.

[9] R. W. G. Hunt. *The Reproduction of Color in Photography, Printing, and Television.* Fountain Press, 3rd edition, 1987.

[10] Keith Jack. *Video Demystified.* HighText Interactive, 2nd edition, 1996.

[11] B-J Kim and W.A. Pearlman. An embedded wavelet video coder using three-dimensional set partitioning in hierarchical trees (SPIHT). In *IEEE Data Compression Conference*, March 1997.

[12] B-J Kim, Z. Xiong, and W.A. Pearlman. Very low bit-rate embedded video coding with 3D set partitioning in hierarchical trees (3D SPIHT). Submitted to IEEE Trans. on Circuits and Systems for Video Technology, October 1997.

[13] G.G. Langdon. An introduction to arithmetic coding. *IBM Journal of Research and Development*, pages 135–149, March 1984.

[14] Stephane Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 674–693, July 1989.

[15] A. Moffat, R. Neal, and I.H. Witten. Arithmetic coding revisited. In *IEEE Data Compression Conference*, pages 202–211, March 1995.

[16] Alistair Moffat, Radford Neal, and Ian H. Witten. Arithmetic coding revisited: Source code. ftp://munnari.oz.au/pub/arith_coder/.

[17] MPEG Software Simulation Group (MSSG). MPEG-2 video codecs (incorporating MPEG-1). http://www.mpeg.org/MMSG/.

[18] K. T. Mullen. The contrast sensitivity of human colour vision to red-green and blue-yellow chromatic gratings. *Journal of Physiology*, pages 381–400, 1985.

[19] Arun N. Netravali and Barry G. Haskell. *Digital Pictures: Representation, Compression, and Standards.* Plenum Press, 2 edition, 1995.

[20] City University of Hong Kong Image Processing Laboratory. Image database. http://www.image.cityu.edu.hk/imagedb/.

[21] William B. Pennebaker and Joan L. Mitchell. *JPEG Still Image Data Compression Standard.* Van Nostrand Reinhold, 1993.

[22] M. Rabbani and P.W. Jones. *Digital Image Compression Techniques.* SPIE Optical Engineering Press, 1991.

[23] Kannan Ramchandran, Martin Vetterli, and Cormac Herley. Wavelets, subband coding, and best bases. In *IEEE Special Issue on Wavelets*, June 1996.

[24] K. Rao and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Academic Press, Inc., 1990.

[25] A. Said and W.A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6, June 1996.

[26] ITU Telecommunication Standardization Sector. Recommendation h.261: Video codec for audiovisual services at $p \times 64$ kbits/s. Technical report, ITU, March 1993.

[27] ITU Telecommunication Standardization Sector. Recommendation h.261: Video coding for low bit rate communication. Technical report, ITU, March 1996.

[28] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12), December 1993.

[29] Eric J. Stollnitz, Tony D. Derose, and Davis H. Salesin. *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kaufmann Publishers, Inc., 1996.

[30] Gilbert Strang and Truong Nguyen. *Wavelets and Filter Banks.* Wellesley-Cambridge Press, 1996.

[31] J.Y. Tham, S. Ranganath, and A.A. Kassim. Highly scalable wavelet-based video codec for very low bit-rate environment. *IEEE Journal on Selected Areas in Communications - Special Issue on Very Low Bit-rate Video Coding*, 16(1):12–27, January 1998.

[32] Mladen Victor Wickerhauser. *Adapted Wavelet Analysis from Theory to Software.* Wellesley, 1994.

[33] B. Wohlberg. Introduction to wavelets. Course Notes, March 1997. Dept. of Electrical Engineering, University of Cape Town.

Appendix A

Colour Space Conversions

The following transformations between the RGB and YC_rC_b colour spaces are taken from Jack [10], and presented here *without* a luminance offset. Several alternative RGB to YC_rC_b colour mappings exist, such as that found in [21].

A.1 RGB to YC_rC_b

$$\begin{aligned} Y &= 0.257R + 0.504G + 0.098B \\ C_r &= 0.439R - 0.368G - 0.071B + 128 \\ C_b &= 0.148R - 0.291G + 0.439B + 128 \end{aligned}$$

A.2 YC_rC_b to RGB

$$\begin{aligned} R &= 1.164Y + 1.596(C_r - 128) \\ G &= 1.164Y + 0.813(C_r - 128) - 0.391(C_b - 128) \\ B &= 1.164Y + 2.018(C_b - 128) \end{aligned}$$

Appendix B

Detailed Results

This Appendix illustrates the frame-by-frame performance of SPRED and MPEG-1 on the seven test-sequences. These results are summarised in Table 7.2. For convenience, a list of the various scenes is provided in the table below. (This list is a repeat of Table 7.1.)

Table B.1: A List of the Scenes in the Test Sequences used

Sequence	Frames	Type of Camera Motion	Translational Object Motion	Degree of Spatial Detail
Claire	1-164	Still	No	Low
Salesman	1-100	Still	Yes	High
Bike	1-148	Pan	No	Medium
Flower Garden	1-112	Translation	No	High
Football 1	1-70	Still	No	Medium
Football 1	71-124	Pan	Yes	High
Football 2	1-84	Still	Yes	Medium
Football 2	85-208	Pan	No	Medium
Table Tennis	1-24	Still	Yes	Low
Table Tennis	25-67	Zoom Out	Yes	Medium
Table Tennis	68-97	Still	No	Low
Table Tennis	98-112	Still	No	Medium

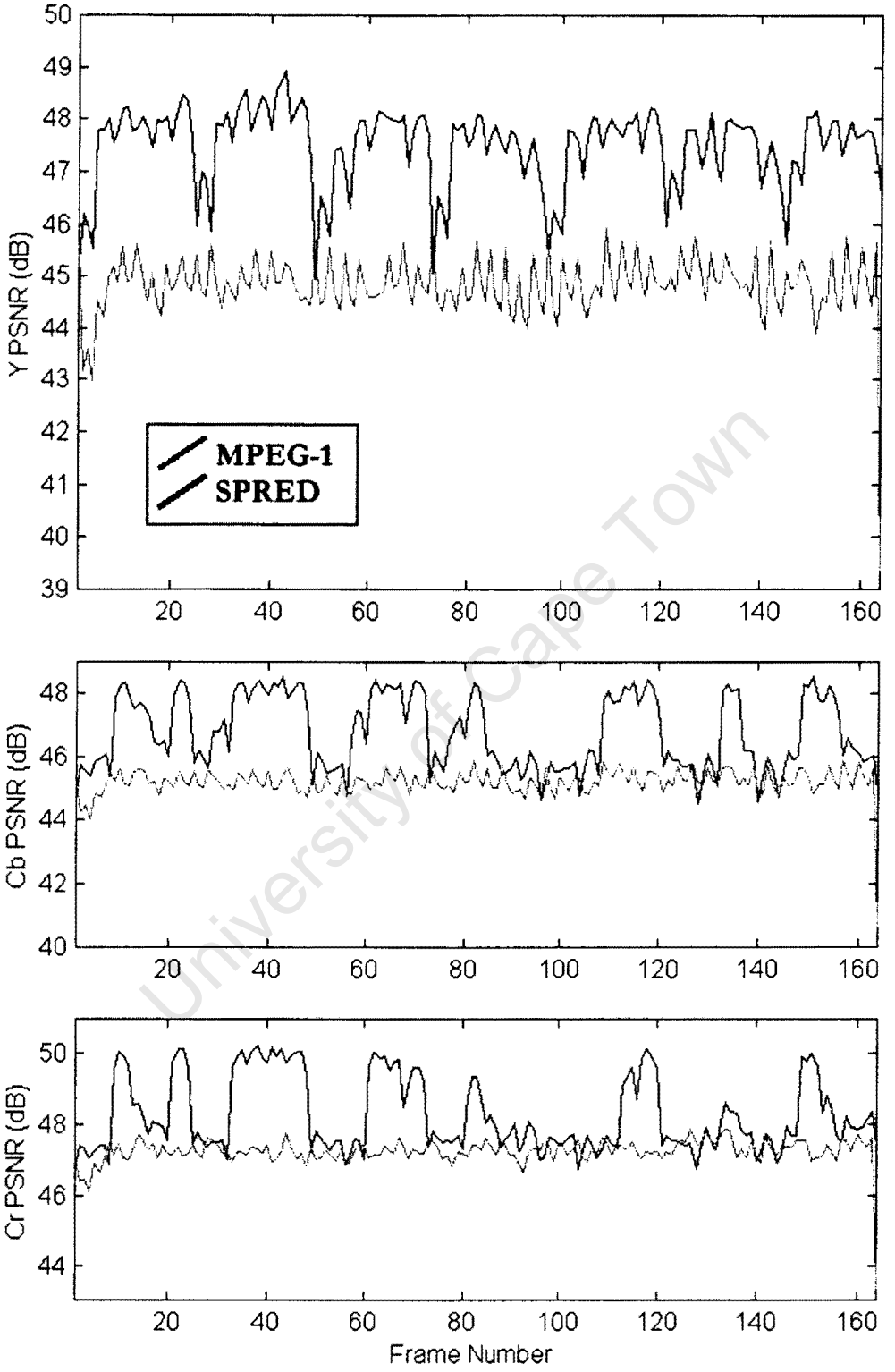


Figure B.1: "Claire" sequence at 30 fps and 1.0 Mbps

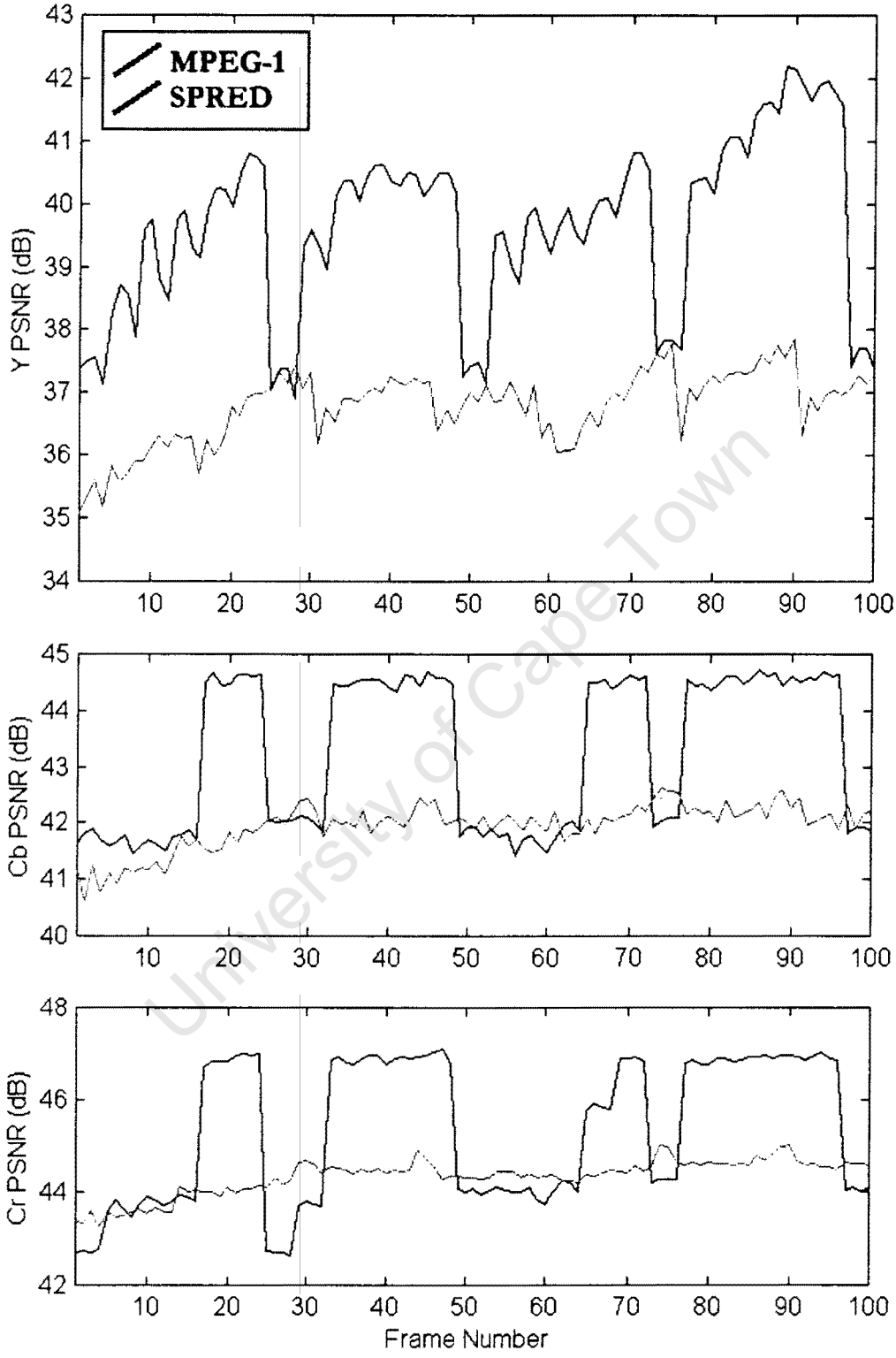


Figure B.2: "Salesman" sequence at 30 fps and 1.0 Mbps

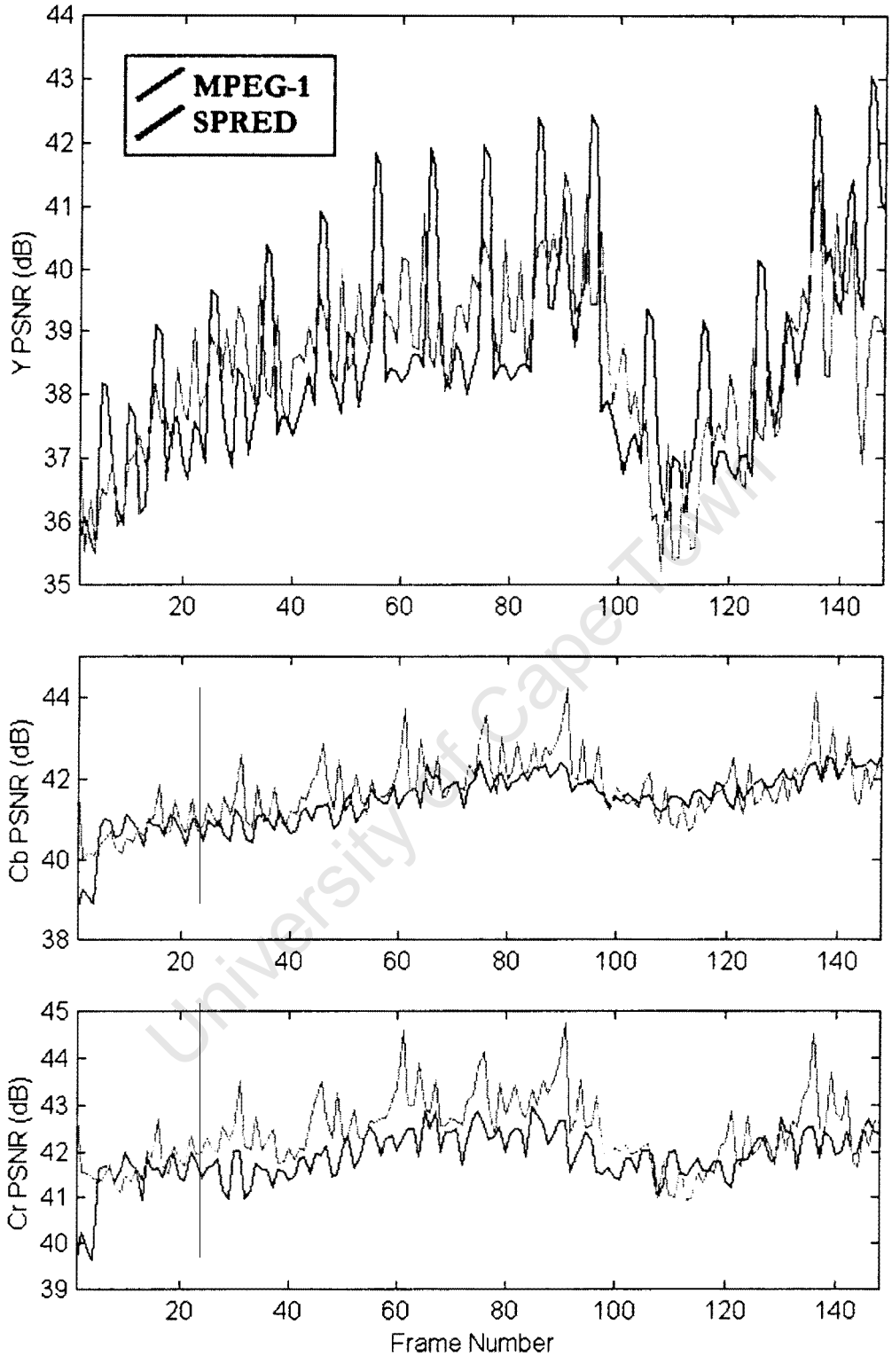


Figure B.3: "Bike" sequence at 30 fps and 1.0 Mbps

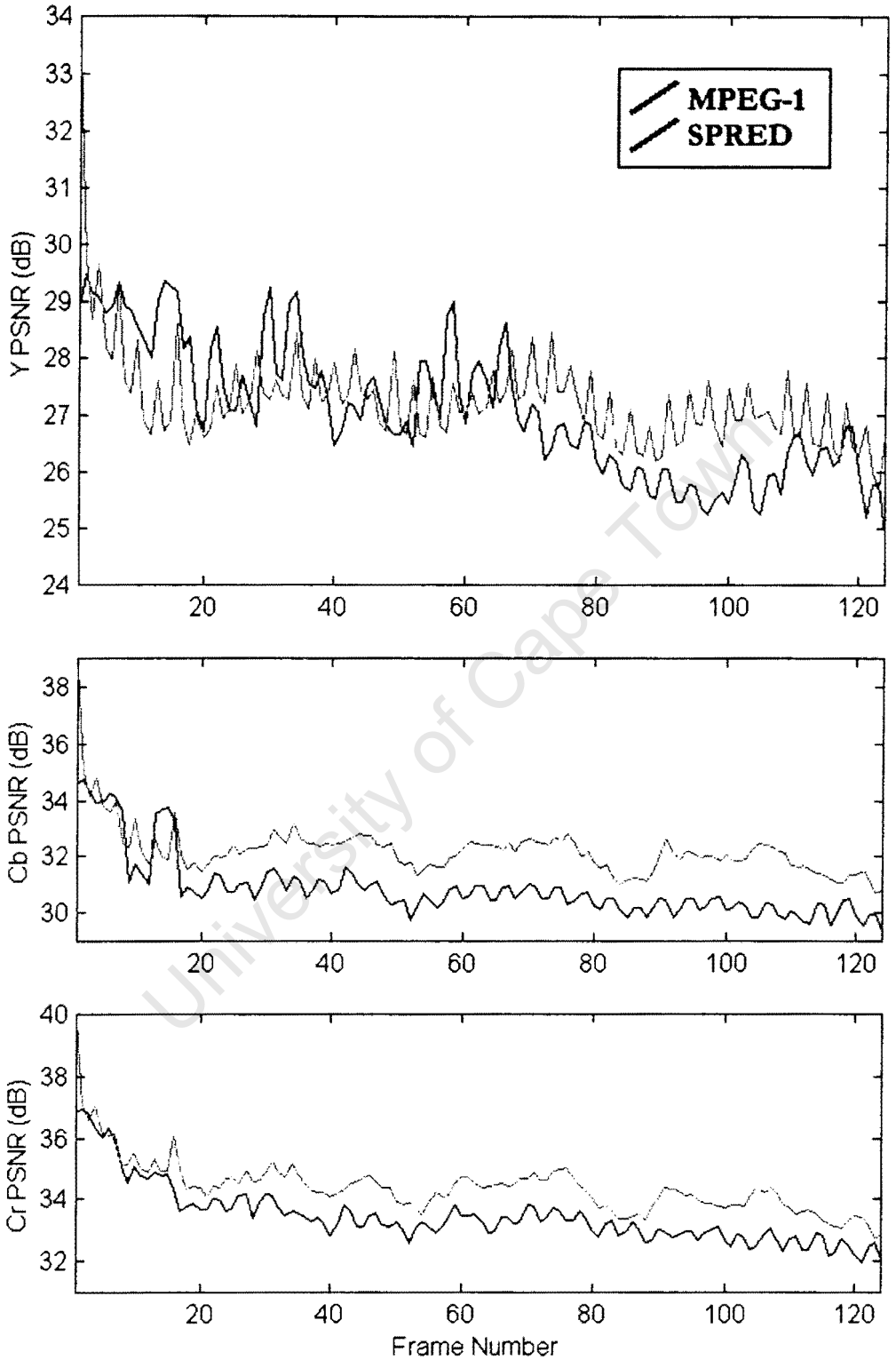


Figure B.4: "Football 1" sequence at 30 fps and 1.0 Mbps

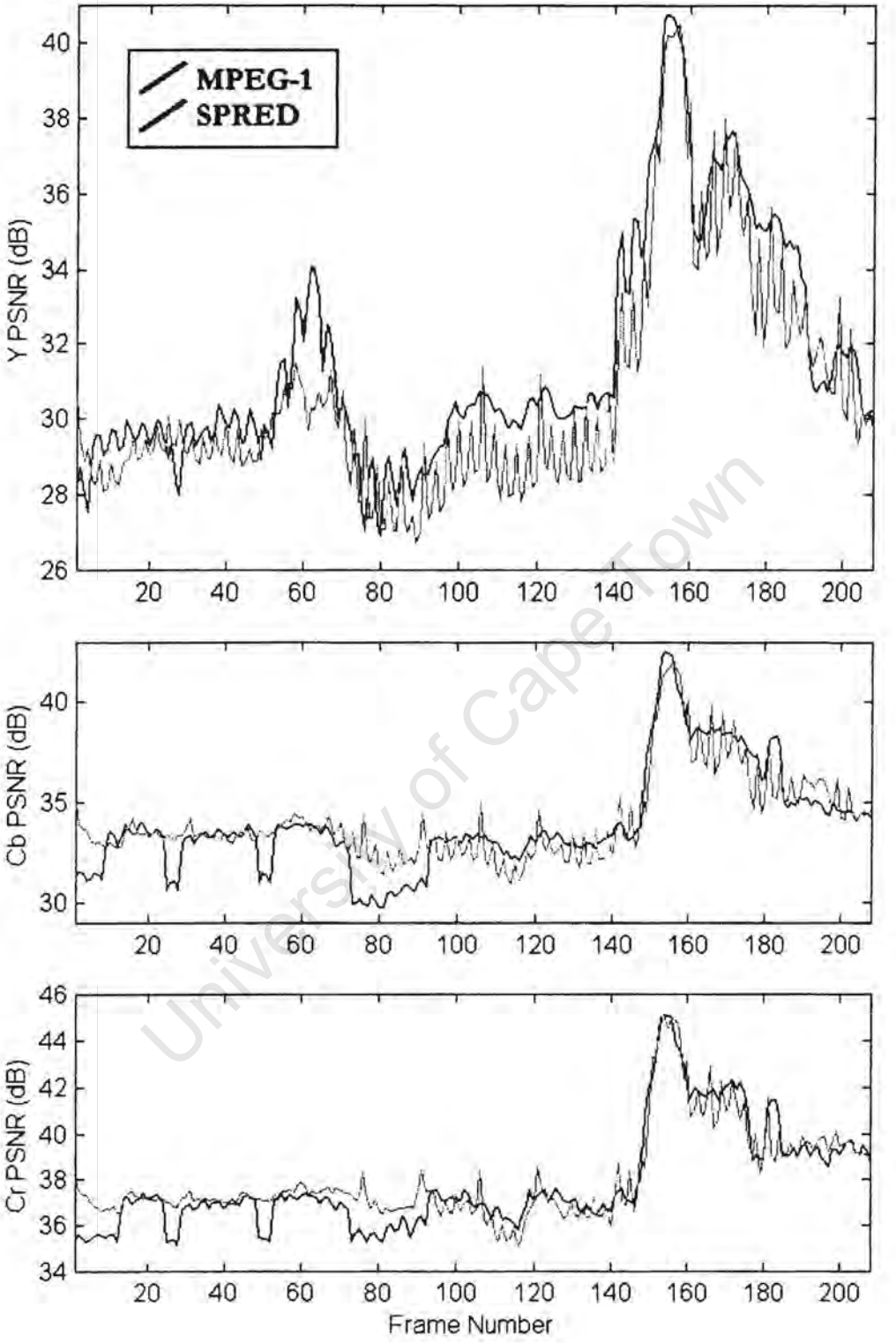


Figure B.5: "Football 2" sequence at 30 fps and 1.0 Mbps

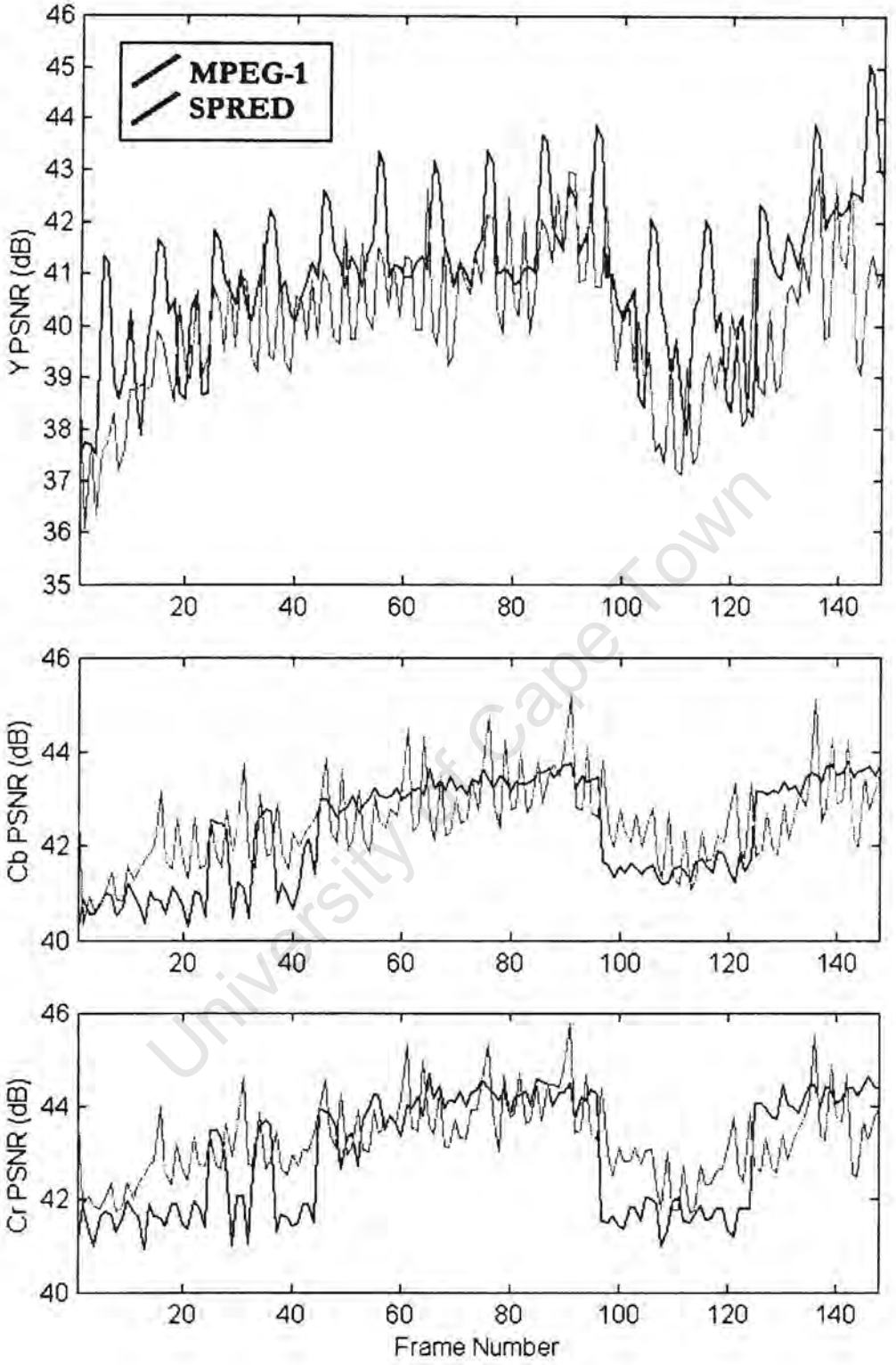


Figure B.6: "Bike" sequence at 30 fps and 1.5 Mbps

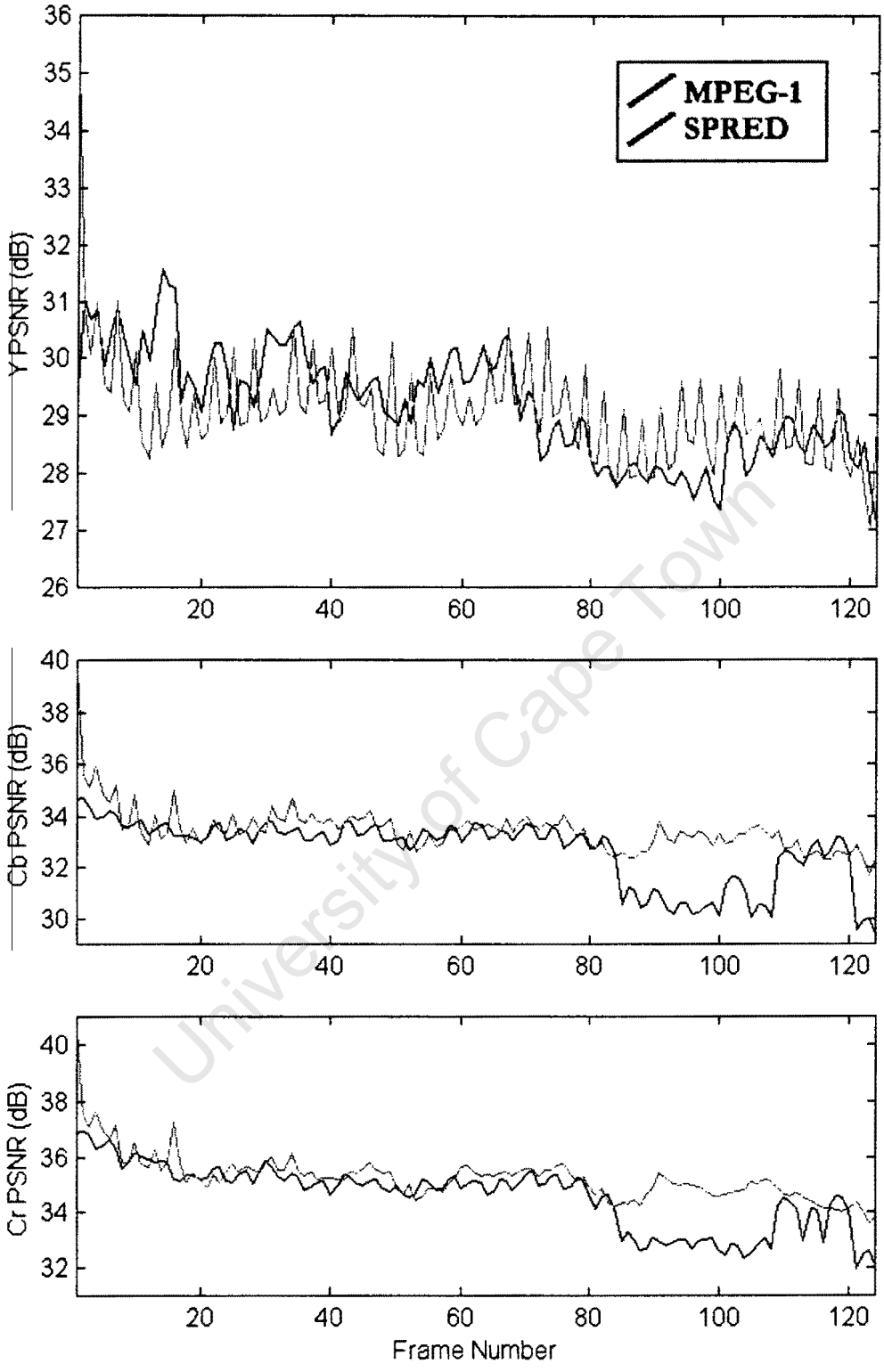


Figure B.7: "Football 1" sequence at 30 fps and 1.5 Mbps

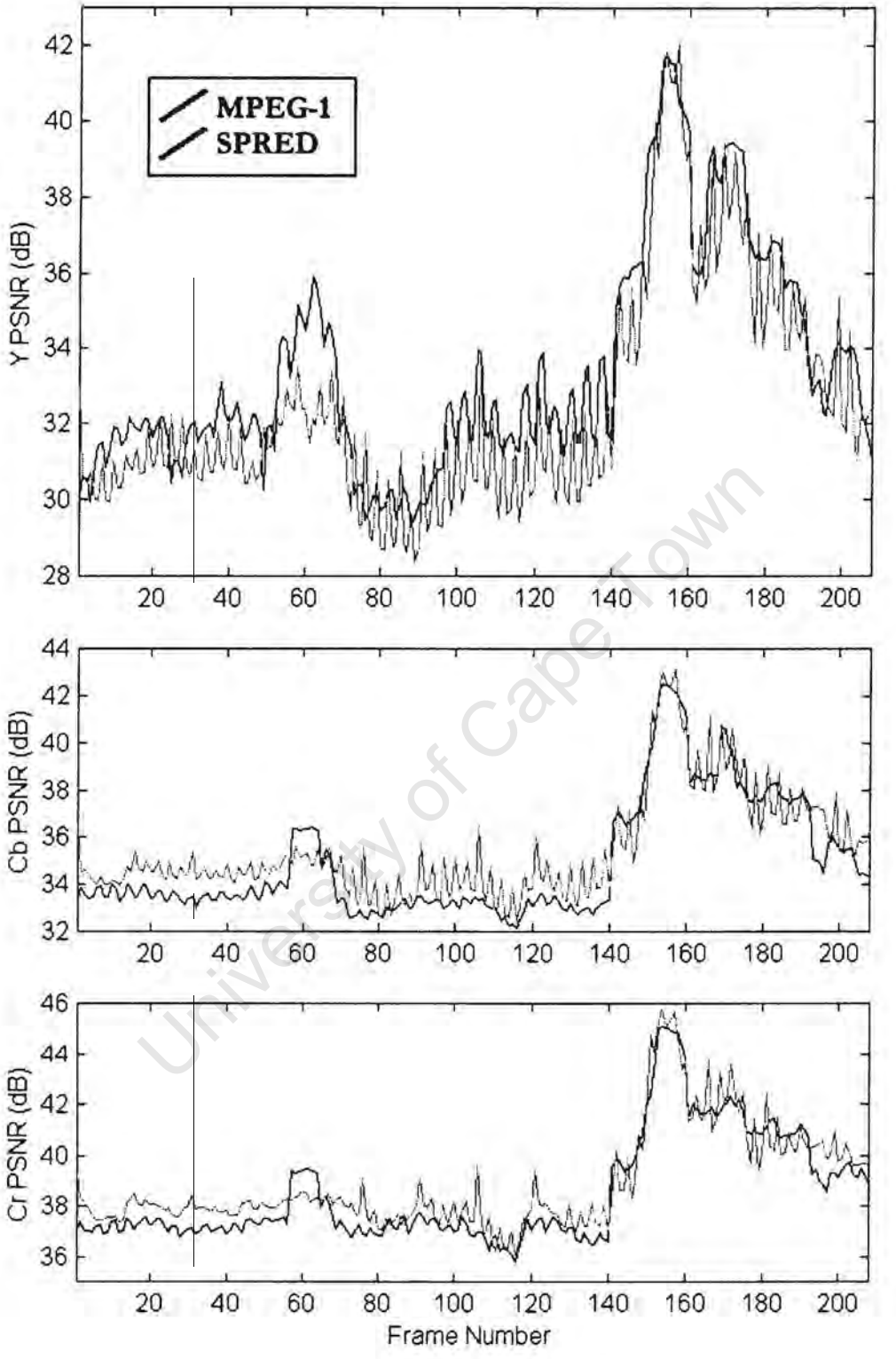


Figure B.8: "Football 2" sequence at 30 fps and 1.5 Mbps

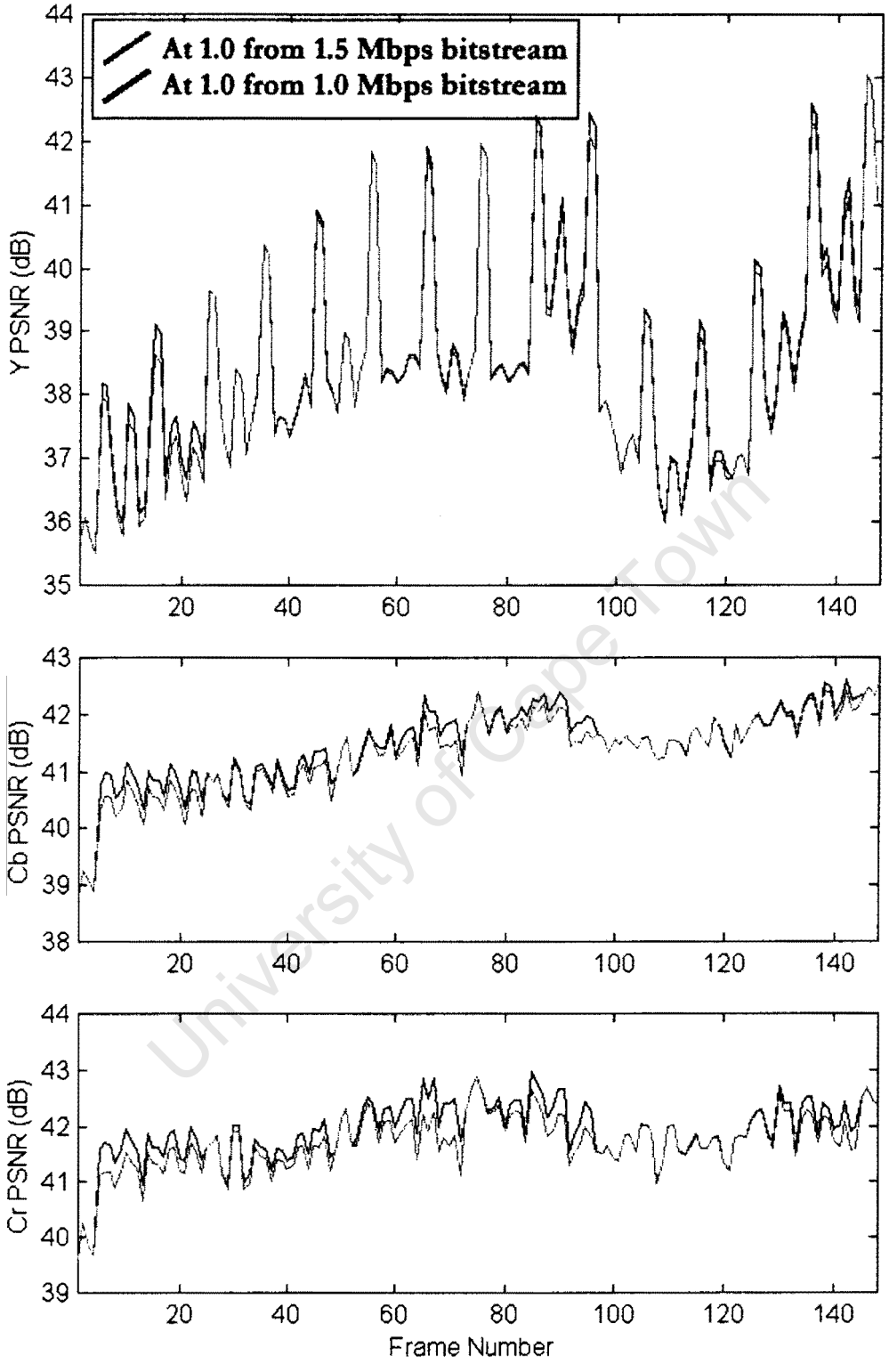


Figure B.9: "Bike" sequence at 30 fps and 1.0 Mbps

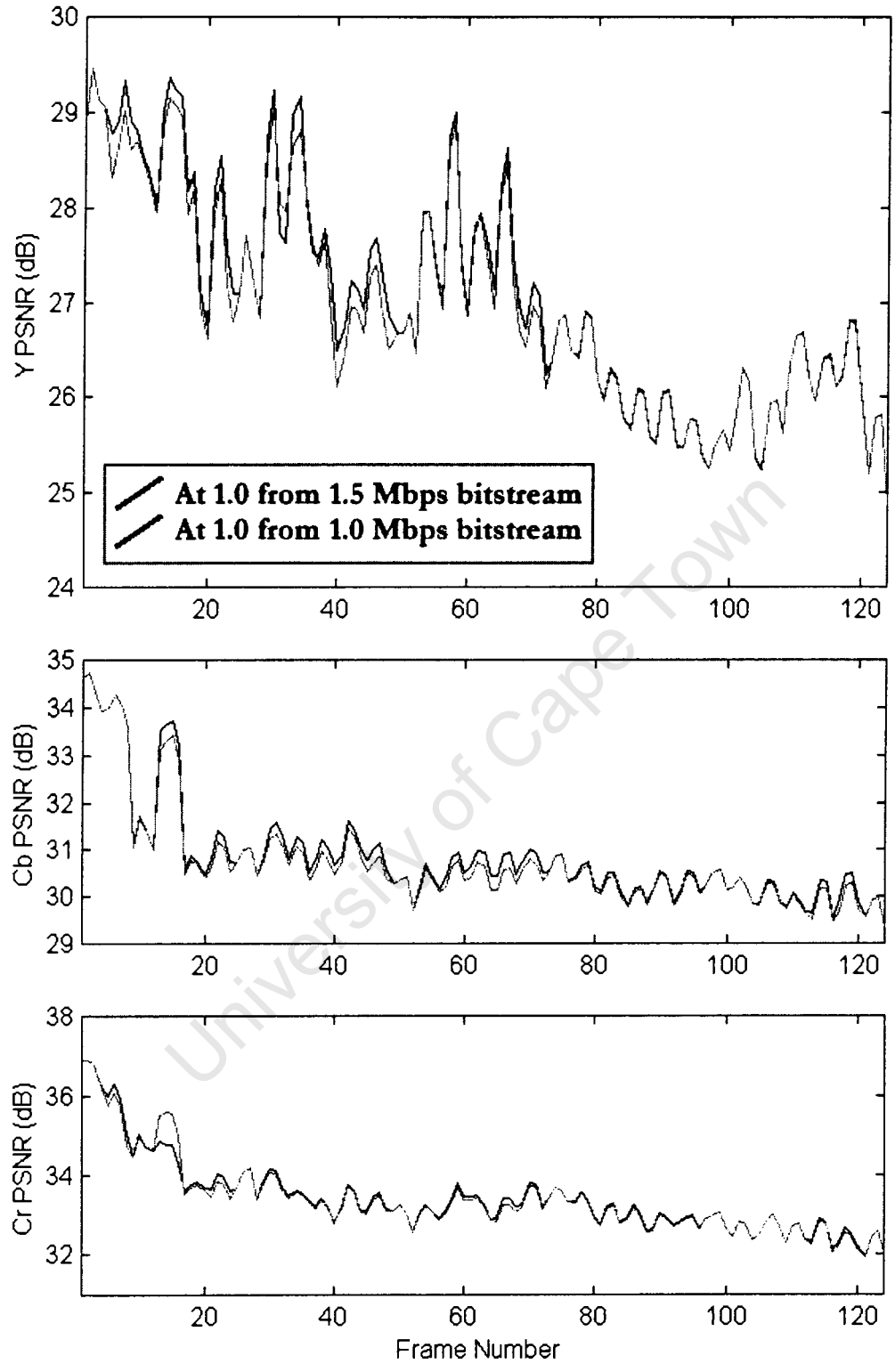


Figure B.10: "Football 1" sequence at 30 fps and 1.0 Mbps

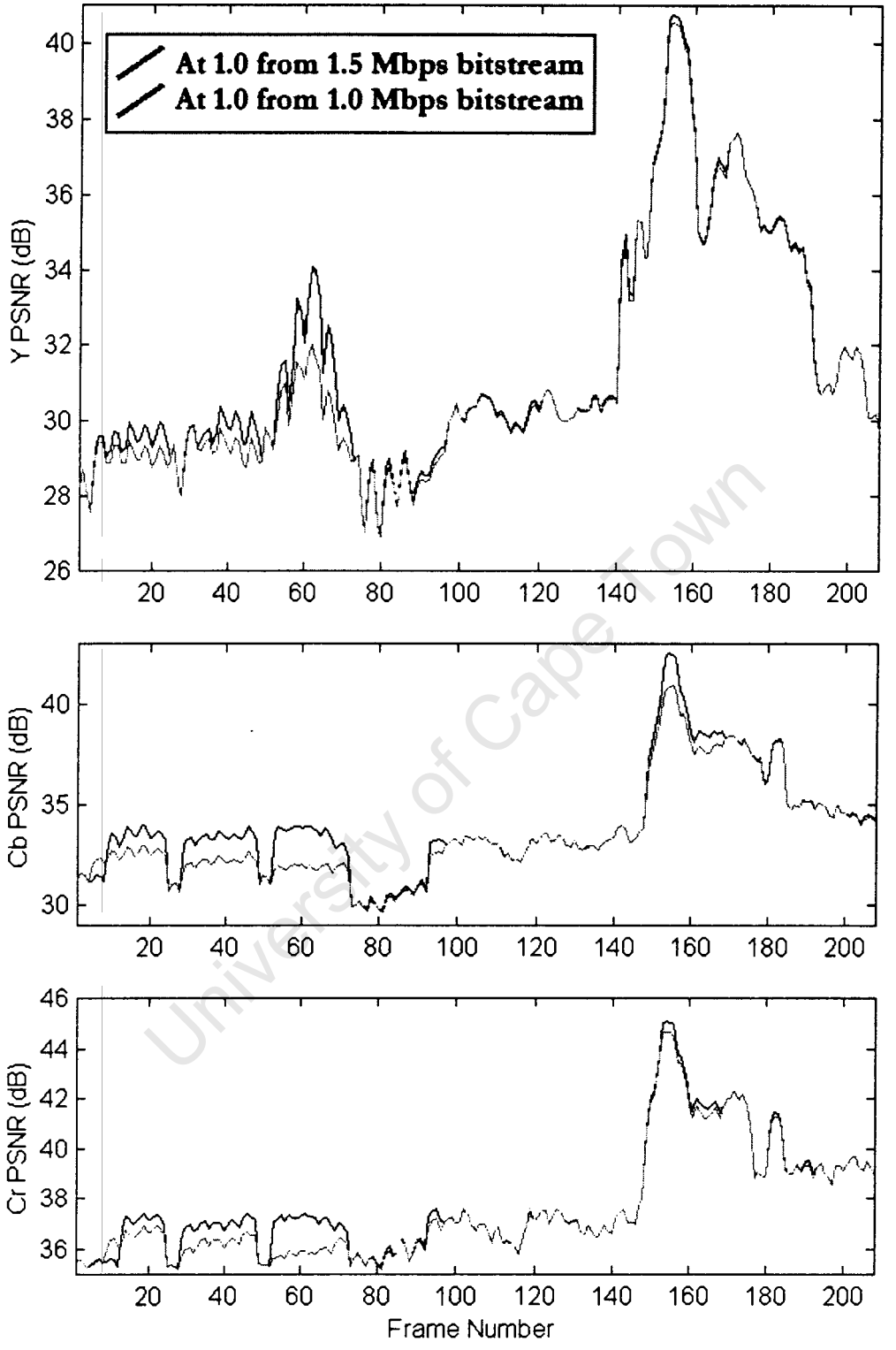


Figure B.11: 'Football 2" sequence at 30 fps and 1.0 bps

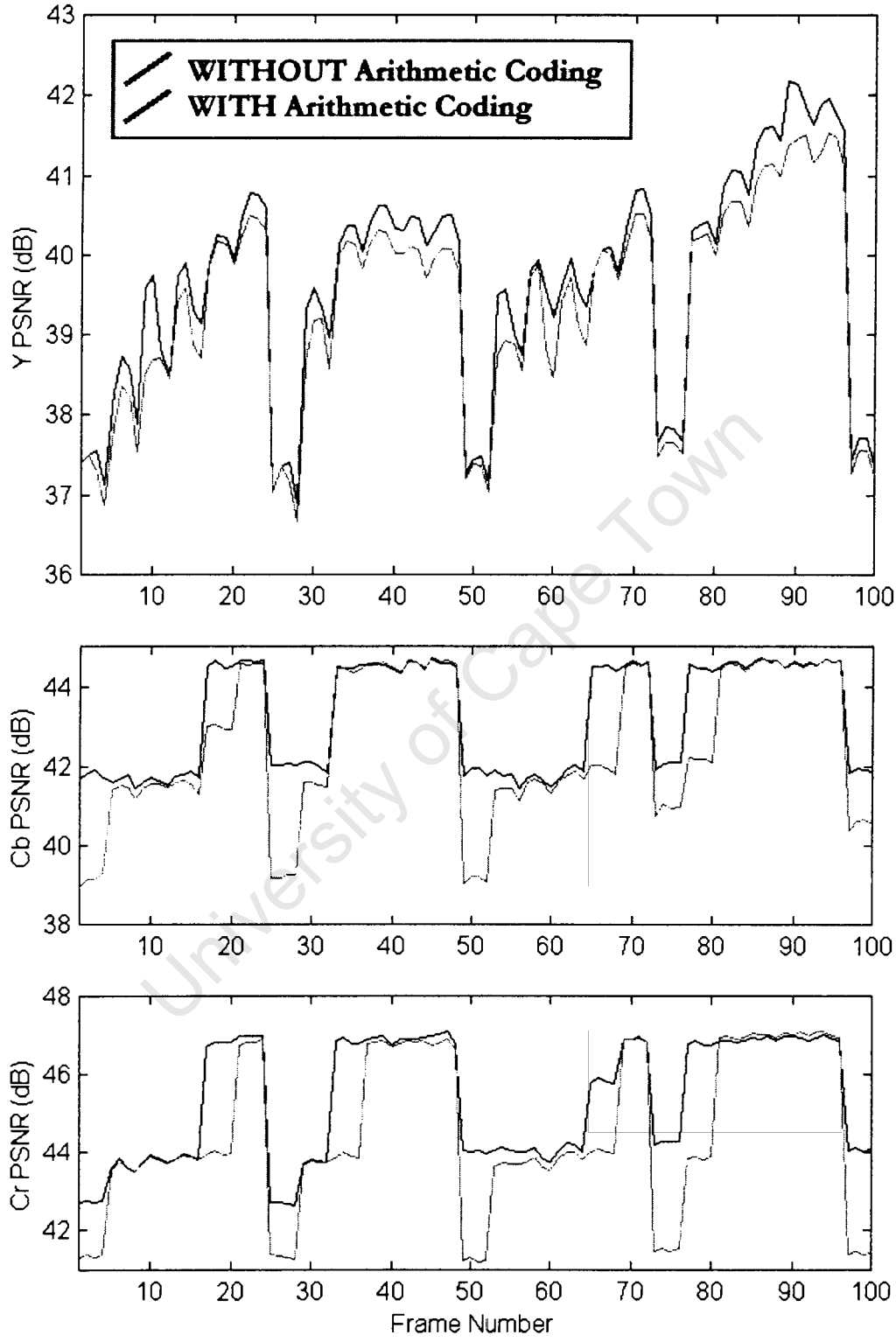


Figure B.12: The "Salesman" sequence at 1.0 Mbps and 30 fps, with and without Binary Arithmetic Coding.

Appendix C

MPEG Compression Parameters

This Appendix provides a list of the options used for the MPEG encoder when compressing the test sequences. The MPEG codec was downloaded from [17]. Note that the codec is able to generate and decode both MPEG-1 and MPEG-2 bit-streams.

"Bike" sequence, compressed with MPEG-1 at 1.5 Mbps and 30 fps:

```
MPEG-2 Test Sequence, 30 frames/sec
D:\Videos\YUV\SIF\bike\pic%03d  /* name of source files */
D:\Videos\YUV\SIF\bike\out%d  /* name of reconstructed images */
-         /* name of intra quant matrix file     ("-": default matrix) */
-         /* name of non intra quant matrix file ("-": default matrix) */
-         /* name of statistics file ("-": stdout ) */
1         /* input picture file format: 0=*.Y,*.U,*.V, 1=*.yuv, 2=*.ppm */
148       /* number of frames */
1         /* number of first frame */
00:00:00:00 /* timecode of first frame */
15        /* N (# of frames in GOP) */
3         /* M (I/P frame distance) */
1         /* ISO/IEC 11172-2 stream */
0         /* 0:frame pictures, 1:field pictures */
352       /* horizontal_size */
240       /* vertical_size */
2         /* aspect_ratio_information 1=square pel, 2=4:3, 3=16:9, 4=2.11:1 */
5         /* frame_rate_code 1=23.976, 2=24, 3=25, 4=29.97, 5=30 frames/sec. */
1572864.0 /* bit_rate (bits/s) */
```

```
20          /* vbv_buffer_size (in multiples of 16 kbit) */
0           /* low_delay  */
0           /* constrained_parameters_flag */
4           /* Profile ID: Simple = 5, Main = 4, SNR = 3, Spatial = 2, High = 1 */
10          /* Level ID:   Low = 10, Main = 8, High 1440 = 6, High = 4          */
1           /* progressive_sequence */
1           /* chroma_format: 1=4:2:0, 2=4:2:2, 3=4:4:4 */
2           /* video_format: 0=comp., 1=PAL, 2=NTSC, 3=SECAM, 4=MAC, 5=unspec. */
5           /* color_primaries */
5           /* transfer_characteristics */
4           /* matrix_coefficients */
352         /* display_horizontal_size */
240         /* display_vertical_size */
0           /* intra_dc_precision (0: 8 bit, 1: 9 bit, 2: 10 bit, 3: 11 bit */
1           /* top_field_first */
0 0 0       /* frame_pred_frame_dct (I P B) */
0 0 0       /* concealment_motion_vectors (I P B) */
1 1 1       /* q_scale_type  (I P B) */
0 0 0       /* intra_vlc_format (I P B)*/
0 0 0       /* alternate_scan (I P B) */
0           /* repeat_first_field */
1           /* progressive_frame */
0           /* P distance between complete intra slice refresh */
0           /* rate control: r (reaction parameter) */
0           /* rate control: avg_act (initial average activity) */
0           /* rate control: Xi (initial I frame global complexity measure) */
0           /* rate control: Xp (initial P frame global complexity measure) */
0           /* rate control: Xb (initial B frame global complexity measure) */
0           /* rate control: d0i (initial I frame virtual buffer fullness) */
0           /* rate control: d0p (initial P frame virtual buffer fullness) */
0           /* rate control: d0b (initial B frame virtual buffer fullness) */
2 2 11 11 /* P:  forw_hor_f_code forw_vert_f_code search_width/height */
1 1 3  3  /* B1: forw_hor_f_code forw_vert_f_code search_width/height */
1 1 7  7  /* B1: back_hor_f_code back_vert_f_code search_width/height */
1 1 7  7  /* B2: forw_hor_f_code forw_vert_f_code search_width/height */
1 1 3  3  /* B2: back_hor_f_code back_vert_f_code search_width/height */
```

"Claire" sequence, compressed with MPEG-2 at 1.0 Mbps and 30 fps:

```
MPEG-1 Test Sequence, 30 frames/sec
D:\Videos\YUV\CIF\claire\pic%03d    /* name of source files */
D:\Videos\YUV\CIF\claire\out%d       /* name of reconstructed images  */
-         /* name of intra quant matrix file     ("-": default matrix) */
-         /* name of non intra quant matrix file ("-": default matrix) */
-         /* name of statistics file ("-": stdout ) */
1         /* input picture file format: 0=*.Y,*.U,*.V, 1=*.yuv, 2=*.ppm */
164       /* number of frames */
1         /* number of first frame */
00:00:00:00 /* timecode of first frame */
15        /* N (# of frames in GOP) */
3         /* M (I/P frame distance) */
0         /* ISO/IEC 11172-2 stream */
0         /* 0:frame pictures, 1:field pictures */
352       /* horizontal_size */
288       /* vertical_size */
2         /* aspect_ratio_information 1=square pel, 2=4:3, 3=16:9, 4=2.11:1 */
5         /* frame_rate_code 1=23.976, 2=24, 3=25, 4=29.97, 5=30 frames/sec. */
1048576.0 /* bit_rate (bits/s) */
20        /* vbv_buffer_size (in multiples of 16 kbit) */
0         /* low_delay  */
0         /* constrained_parameters_flag */
4         /* Profile ID: Simple = 5, Main = 4, SNR = 3, Spatial = 2, High = 1 */
10        /* Level ID:   Low = 10, Main = 8, High 1440 = 6, High = 4          */
1         /* progressive_sequence */
1         /* chroma_format: 1=4:2:0, 2=4:2:2, 3=4:4:4 */
2         /* video_format: 0=comp., 1=PAL, 2=NTSC, 3=SECAM, 4=MAC, 5=unspec. */
5         /* color_primaries */
5         /* transfer_characteristics */
4         /* matrix_coefficients */
352       /* display_horizontal_size */
288       /* display_vertical_size */
0         /* intra_dc_precision (0: 8 bit, 1: 9 bit, 2: 10 bit, 3: 11 bit */
1         /* top_field_first */
0 0 0     /* frame_pred_frame_dct (I P B) */
0 0 0     /* concealment_motion_vectors (I P B) */
1 1 1     /* q_scale_type  (I P B) */
```

```
1 0 0     /* intra_vlc_format (I P B)*/
0 0 0     /* alternate_scan (I P B) */
0         /* repeat_first_field */
1         /* progressive_frame */
0         /* P distance between complete intra slice refresh */
0         /* rate control: r (reaction parameter) */
0         /* rate control: avg_act (initial average activity) */
0         /* rate control: Xi (initial I frame global complexity measure) */
0         /* rate control: Xp (initial P frame global complexity measure) */
0         /* rate control: Xb (initial B frame global complexity measure) */
0         /* rate control: d0i (initial I frame virtual buffer fullness) */
0         /* rate control: d0p (initial P frame virtual buffer fullness) */
0         /* rate control: d0b (initial B frame virtual buffer fullness) */
2 2 11 11 /* P:  forw_hor_f_code forw_vert_f_code search_width/height */
1 1 3  3  /* B1: forw_hor_f_code forw_vert_f_code search_width/height */
1 1 7  7  /* B1: back_hor_f_code back_vert_f_code search_width/height */
1 1 7  7  /* B2: forw_hor_f_code forw_vert_f_code search_width/height */
1 1 3  3  /* B2: back_hor_f_code back_vert_f_code search_width/height */
```